Jaakko Virta

Taajuusmuuttajan järjestelmätestauksen kehittäminen

Sähkötekniikan korkeakoulu

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 8.1.2015.

Työn valvoja:

Prof. Raimo Sepponen

Työn ohjaaja:

DI Ville Eskola

AALTO-YLIOPISTO DIPLOMITYÖN

SÄHKÖTEKNIIKAN KORKEAKOULU TIIVISTELMÄ

Tekijä: Jaakko Virta

Työn nimi: Taajuusmuuttajan järjestelmätestauksen kehittäminen

Päivämäärä: 8.1.2015 Kieli: Suomi Sivumäärä: 6+56

Sähkötekniikan ja automaation laitos

Professuuri: Elektroniikka ja sovellukset Koodi: S-66

Valvoja: Prof. Raimo Sepponen

Ohjaaja: DI Ville Eskola

Tämän diplomityön tarkoitus on kehittää taajuusmuuttajan järjestelmätestausvaihetta ABB Oy:n High Power Drives -osastolla. Työssä on selvitetty järjestelmätestauksen osa-alueet ja sen rooli järjestelmän kehityksessä. Kirjallisuuskatsauksen ja asiantuntijahaastatteluiden perusteella on pohdittu kehityskohteita järjestelmätestausvaiheen kattavuudessa. Järjestelmätestauksen tehostamiseksi on työssä kehitetty automaattitestausta, jonka avulla on tarkoitus lisätä käytössä olevaa testausaikaa ja vapauttaa henkilöstöresursseja muihin testaustehtäviin.

Diplomityön tuloksena on toteutettu automaattitestausjärjestely, joka kykenee ajamaan taajuusmuuttajan tehokuormituspisteitä, valvomaan testin kulkua sekä tallentamaan testin aikana kerättyjä mittaustuloksia. Testausjärjestely on kehitetty valmiina olleen ratkaisun pohjalle sekä koestettu ja käyttöönotettu alustavasti lyhyillä testiajoilla. Työssä esitetään järjestelyn toiminta ja mahdolliset kehitystarpeet sen toiminnallisuudessa ja käytettävyydessä.

Testauksen kattavuuden parantamiseksi on työssä esitetty useita suorituskykyä ja taajuusmuuttajan toimintoja testaavia testitilanteita, joiden perusteella voidaan laajentaa nykyistä järjestelmätestaussuunnitelmaa. Testien yksityiskohtainen suunnittelu vaatii kuitenkin lisätutkimusta.

Avainsanat: Järjestelmätestaus, automaattitestaus, taajuusmuuttaja, testauksen kehitys

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Jaakko Virta

Title: Improving the system testing of a frequency converter

Date: 8.1.2015 Language: Finnish Number of pages: 6+56

Department of Electrical Engineering and Automation

Professorship: Electronics and Applications Code: S-66

Supervisor: Prof. Raimo Sepponen

Advisor: M.Sc. Ville Eskola

The purpose of this thesis is to improve the system test phase of a frequency converter in High Power Drives department of ABB Oy. The fields included in system testing and the role of system test phase in system development are considered. Based on a literary review and information gained by expert interviews, some targets for improvement in system test phase coverage are thought of. Automated testing has been developed to enhance system testing in practice. The object of automated testing is to increase available testing time and to free personnel to other fields of testing.

As a result of this thesis an automated testing setup has been built. The setup can execute and supervise a load test of the frequency converter and save measurement data for further use. The test setup has been implemented on the basis of a former automated test solution and the setup has been tested and commissioned with short test runs. The operation of the test setup is presented in this work, and possible improvements in the functionality and usability of the setup are considered.

Many possibilities for the improvement of test coverage are suggested in this thesis. The suggestions aim to improve the testing of functionality and performance of the frequency converter in system test phase. However, the detailed test planning needs further research.

Keywords: System testing, automated testing, frequency converter, test improvement

Esipuhe

Tämä diplomityö on tehty ABB Oy Drivesin High Power Drives -osastolle. Työn tekemisen on mahdollistanut useiden ihmisten apu. Kiitän Ville Eskolaa työn ohjaamisesta ja etenkin neuvoista käytännön osuuden toteuttamisessa. Kiitos myös valvojalleni, professori Raimo Sepposelle, työn tarkastuksesta ja neuvoista työn rakenteen ja kirjoitusasun suhteen. Työni aikana olen saanut apua myös useilta muilta ABB Drivesin tuotekehityksessä työskenteleviltä työtovereiltani, joita haluan myös kiittää kaikesta avusta.

Kiitokset myös vanhemmilleni ja sisaruksilleni kaikesta tuesta opiskelujen varrella sekä Susannalle kannustuksesta ja kärsivällisyydestä diplomityön tekemisen aikana.

Espoo, 8.1.2015

Jaakko J. Virta

Sisältö

Symbolit ja lyhenteet

ABB	Asea Brown Boveri, monialainen kansainvälinen teknologiayritys
DTC	Direct Torque Control, ABB:n kehittämä momentin suoraan ohjaukseen perustuva taajuusmuuttajan ohjaustapa
FBD	Function Block Diagram, IEC61131-3 standardin mukainen lohkokaaviomuotoinen ohjelmointikieli
HPD	High Power Drives, ABB Oy:n korkeatehoisista taajuusmuuttajista vastaava osasto
HTR	Heat Test Runner, ABB Oy:ssä kehitetty kuormitettuihin lämpötesteihin tarkoitettu automaattinen testausympäristö
IGBT	Insulated Gate Bipolar Transistor. Suuritehoisiin sovelluksiin tarkoitettu bipolaaritransistori, jonka hila on eristetty kollektorista ja emitteristä
LAC	Low Power AC Drives, ABB Oy:n matalatehoisista taajuusmuuttajista vastaava osasto
PLC	Programmable Logic Controller, Ohjelmoitava logiikka
ST	Structured Text, IEC61131-3 standardin mukainen tekstimuotoinen ohjelmointikieli

1 Johdanto

Testaus on oleellinen osa tuotekehitystä. Teollisuuden tuotekehityksessä testaus kulkee jatkuvasti suunnittelun rinnalla ja jokaisessa kehitysvaiheessa on omat testaustoimenpiteet, jotka tukevat tuotteen valmistumista.

Tässä diplomityössä tutkitaan tuotekehityksen järjestelmätestausta. Järjestelmätestaus on testausvaihe, joka aloitetaan kun kehitetyn järjestelmän osat toimivat itsenäisesti ja on integroitu keskenään onnistuneesti. Testaus keskittyy tässä vaiheessa etsimään vikoja järjestelmästä kokonaisuutena.

Työ on tehty ABB:n Drives -yksikön alaiselle High Power Drives (HPD) -osastolle Helsingin Pitäjänmäen tehtaalla. ABB on monikansallinen yritys, jonka toimialat kattavat useita sähköalan osa-alueita. Pitäjänmäellä yritys on keskittynyt sähkömoottoreihin ja -generaattoreihin sekä taajuusmuuttajiin. HPD:n vastuualue on suuritehoisten taajuusmuuttajien kehittäminen. Tällä hetkellä High Power Drivesin tuotekehitys suunnittelee ACS880-tuoteperheen taajuusmuuttajia.

Diplomityön tavoitteena on selvittää tapoja, joilla HPD:n suunnitteleman taajuusmuuttajan järjestelmätestausta voisi kehittää. Työssä selvitetään testauksen organisointi osastolla aikaisemmin ja etsitään järjestelmätestausvaiheessa kehitettäviä osa-alueita. Taajuusmuuttajan suunnitteluprosessiin ja testausvastuisiin on perehdytty ABB:n tarjoamien dokumenttien ja eri taajuusmuuttajan osa-alueiden suunnittelusta vastaavien asiantuntijoiden avulla.

Työn varhaisessa vaiheessa on päätetty tehostaa nykyistä järjestelmätestausta automatisoimalla joitakin testejä. Diplomityössä käyttöönotetaan HPD:llä automaattinen testilaitteisto, jonka avulla taajuusmuuttajaa kyetään testaamaan ilman välitöntä testinvalvontaa. Automaattinen testilaitteisto perustuu ABB Drivesin Low Power AC Drives (LAC) -osaston käyttämään testausjärjestelmään, joka koostuu taajuusmuuttajaa ohjaavasta HTR-testiympäristöstä ja mittalaitedataa keräävästä ABB:n AC500 ohjelmoitavasta logiikasta (Programmable Logic Controller, PLC). Työssä muokataan AC500:n Codesys-ohjelmointiympäristössä toteutettua logiikkaohjelmaa vastaamaan uuden mittaustilanteen tarpeita ja käyttöönotetaan testausjärjestelmä valitulla testauspaikalla. HTR-testiympäristöä ei työn puitteissa muokata, vaan pelkästään käyttöönotetaan. Testilaitteiston toiminnan lisäksi otetaan huomioon turvallisuuskysymykset, jotka ovat tärkeitä kun testejä ajetaan ilman valvontaa.

Työn toisessa luvussa esitellään järjestelmäsuunnittelua. Kolmannessa luvussa perehdytään järjestelmän testauksen eri puoliin. Neljännessä luvussa tutkitaan testausmenetelmiä ja viidennessä luvussa testilaitteiston suunnitteluun liittyviä tekijöitä. Kuudennessa luvussa selvitetään järjestelmätestausvaiheessa toteutettavia testejä. Seitsemännessä luvussa esitellään taajuusmuuttaja järjestelmäsuunnittelun kannalta ja kahdeksannessa luvussa pohditaan tapoja kehittää HPD:lla tapahtuvaa järjestelmätestausta. Automaattitestilaitteisto ja sen käyttöönotto esitellään luvussa yhdeksän. Luvuissa kymmenen ja yksitoista esitetään diplomityön aikana tehdyt johtopäätökset, mahdollisuudet jatkokehitykselle sekä yhteenveto tehdystä työstä.

2 Järjestelmäsuunnittelu

Ennen kuin järjestelmätestausta voidaan käsitellä, pitää selvittää mikä suunniteltava järjestelmä on, millaisia testausvaiheita järjestelmän kehitykseen on aiemmin sisältynyt ja mitä kehitettävää aikaisemmissa testausmenetelmissä on. Tässä luvussa määritellään yleisellä tasolla järjestelmäsuunnittelun vaiheet, minkä avulla voidaan ymmärtää järjestelmätestauksen rooli järjestelmän kehitystyössä paremmin.

Järjestelmällä voidaan tarkoittaa mitä tahansa eri osista koostuvaa kokonaisuutta, jonka osien välillä sekä osien ja ympäristön välillä on rajapintoja. Järjestelmä vaikuttaa ympäristöönsä ja on ympäristönsä vaikuttama sisään- ja ulostulojensa välityksellä. [1] Järjestelmän voi jakaa toimintansa perusteella osiin, joita kutsutaan alijärjestelmiksi. Alijärjestelmät ovat yleensä selkeästi määriteltävissä toimintojen tai fyysisten kokonaisuuksien perusteella. Laitteen alijärjestelmiä voivat olla yleisellä tasolla laitteisto ja ohjelmisto ja niiden alijärjestelmiä ohjelmiston eri aliohjelmat ja funktiot sekä laitteiston piirilevyt tai komponentit. Konkreettisesti järjestelmään sisältyvien alijärjestelmien lisäksi järjestelmään vaikuttavat useat järjestelmän toiminnan mahdollistavat komponentit kuten kehitystyökalut, tuotannon työkalut ja huoltotyökalut [1]. Myös nämä tulee ottaa huomioon järjestelmää suunniteltaessa.

Järjestelmän suunnittelemisen voi katsoa koostuvan sarjasta vaiheita, joiden aikana järjestelmä kehittyy konseptista valmiiksi tuotteeksi. Järjestelmäsuunnittelun vaiheita ovat määrittely, suunnittelu, toteutus ja integrointi. Koko järjestelmän onnistuneen integroinnin jälkeen järjestelmä on valmis hyväksyntävaiheeseen. Teoriassa kehitysjärjestys toimii kronologisesti, mutta todellisuudessa järjestelmän eri osia suunnitellaan rinnakkain, ja eri alijärjestelmät voivat olla keskenään samanaikaisesti hyvinkin erilaisissa kehitysvaiheissa. Jokaisessa järjestelmän kehitysvaiheessa on mahdollisuus vikojen syntymiselle, ja tämän takia järjestelmän toimintaa varmistetaan ja järjestelmää testataan koko sen kehityksen ajan. [1], [2]

Kuva 1: Järjestelmän suunnitteluvaiheet. [2, s. 114 mukaillen]

Kuvassa 1 on esitetty yllä mainitut järjestelmän kehitysvaiheet järjestyksessä. Kuvasta voidaan nähdä kuinka jokaisen vaiheen lopputuloksena saadaan perusta seuraavaan vaiheeseen, kunnes tuotekehityksen lopputuloksena käyttäjälle on tarjolla valmis tuote. Joissakin tuotekehitysprosesseissa järjestelmän kehitysvaiheiden väliset tulokset ovat tarkasti määriteltyjä dokumentteja, ja vasta näiden dokumenttien valmistumisen ja katselmoinnin jälkeen kehitysvaihe voidaan todeta päättyneeksi. Kukin kehitysvaihe sisältää toimintoja, joiden huolellinen toteuttaminen varmistaa seuraavienkin vaiheiden onnistumista.

Kehitystyö alkaa järjestelmän määrittelystä. Määrittelyvaiheen aikana luodaan kehitettävälle järjestelmälle yleisluontoiset vaatimukset, jotka luovat perustan järjestelmän suunnittelutyölle. Määrittelyssä täytyy ottaa huomioon ominaisuuksien toteuttamiskelpoisuus, kustannukset ja mahdolliset vaatimusten keskinäiset ristiriidat. Vaatimukset voidaan esittää esimerkiksi kirjallisten raporttien, konseptia kuvaavien mallien ja prototyyppien muodossa. Vaiheen päämääränä on luoda järjestelmän toiminnalle täsmällinen kuvaus, josta ilmenee mikä on suunniteltavan järjestelmän tarkoitus, millä tavoilla se tarkoitustansa toteuttaa ja mitä se ei saa tehdä. Vaiheen lopputuloksena järjestelmän pitäisi olla määritelty mustana laatikkona, jonka ulkoinen toiminta tunnetaan mutta sisäinen tehtävänjako on vielä selvittämättä. [1]

Määrittelyvaiheen jälkeen alkaa suunnitteluvaihe, jonka aikana pyritään suunnittelemaan järjestelmän tekninen toteutustapa ja arkkitehtuuri toteutusta varten. Viimeistään tässä vaiheessa päätetään, mitkä toiminnot toteutetaan laitteistolla ja mitkä ohjelmistolla. Järjestelmä jaetaan hallittaviin alijärjestelmiin, joiden komponentit ja toiminta kuvataan. Myös alijärjestelmien rajapinnat määritellään tarkasti, jotta jatkokehitys voidaan toteuttaa tarpeen vaatiessa modulaarisesti alijärjestelmä kerrallaan. Vaiheen jälkeen suunniteltavan järjestelmän pitäisi olla valmis toteutettavaksi. [1]

Suunnittelun jälkeen järjestelmä toteutetaan. Toteutusvaiheessa järjestelmän alijärjestelmät rakennetaan rajapintaspesifikaatioidensa mukaisesti, ja alijärjestelmien itsenäinen toiminta varmistetaan. [1] Vaiheen päätyttyä on suunnitteluorganisaatiolla ideaalisesti käsissään joukko toimivia alijärjestelmiä, jotka täytyy vain liittää yhteen.

Integrointivaiheessa toteutetut alijärjestelmät kootaan kokonaiseksi järjestelmäksi ja niiden keskinäinen toiminta varmistetaan. [1] Vaiheen haastavuus riippuu suunnittelun tasosta, mutta todennäköisesti alijärjestelmiä ja koko järjestelmän toteutusta täytyy vielä tässä vaiheessa muokata, jotta järjestelmän toiminta olisi paras mahdollinen.

Kun järjestelmää on integrointivaiheessa paranneltu tarpeeksi, on järjestelmä valmis hyväksyntävaiheeseen. Tässä vaiheessa keskitytään valmiin järjestelmän toiminnan osoittamiseen kehittävän organisaation lisäksi standardointilaitosten, sidosryhmien, asiakkaiden ja muiden ulkopuolisten tahojen vaatimusten perusteella. Tässä kehitysvaiheessa ei järjestelmän tulisi enää tarvita muutoksia. Todellisuudessa järjestelmää saatetaan kuitenkin joutua muokkaamaan odottamattomien virheiden seurauksena. Hyväksyntävaiheen lopussa järjestelmä todetaan hyväksytyksi ja sitä voidaan alkaa valmistaa tuotannossa. [1]

Testausorganisaatiolla on roolinsa jokaisessa järjestelmän tuotekehitysvaiheessa [2]. Seuraavaksi perehdytään tarkemmin järjestelmän tuotekehityksen aikaiseen testaukseen ja sen menetelmiin. Testauksen teoriaa tutkiessa perehdytään erityisesti järjestelmätestaukseen liittyviin menetelmiin, jotka liittyvät oleellisesti tämän diplomityön aiheeseen.

3 Järjestelmän testaaminen

Testaus määritellään IEEE:n standardissa 829 toimintana, jossa järjestelmää tai sen osaa käytetään määrätyissä olosuhteissa ja toteutetun tilanteen seurauksia tarkkaillaan ja tallennetaan, minkä jälkeen testauksen kohteen jostain piirteestä tehdään johtopäätöksiä [3]. Järjestelmän käytännön testaus voidaan aloittaa toteutusvaiheessa, kun suunniteltujen laitteistoalijärjestelmien komponentteja valitaan ja ohjelmistoalijärjestelmien funktioiden ensimmäiset versiot valmistuvat. Tätä ennen testausta valmistellaan ja suunnitellaan siinä missä järjestelmääkin.

Järjestelmän suunnitteluvaiheen aikana voidaan kehittää myös testaussuunnitelmaa, jossa määritellään eri komponenteille ja alijärjestelmille tarvittavat testit ja kartoitetaan alustavasti integrointi- ja hyväksyntävaiheen testaustarvetta. Testauksen suunnittelu jatkuu tämän suunnitelman pohjalta koko järjestelmän kehityksen ajan. Jopa järjestelmän hyväksyntävaiheen aikana täytyy testausorganisaation valita ja kehittää tuotekehityksen testausohjelmasta testejä, joita voidaan hyödyntää järjestelmän tuotannon ja ylläpidon aikana. [1]

Testejä suunniteltaessa pitää selvittää järjestelmän määrittelyn avulla, mitä testauksen kohteena olevan järjestelmän osan on tarkoitus tehdä ja mitä se ei missään tapauksessa saa tehdä. Tämän jälkeen kunkin tilanteen testit määritellään täsmällisesti ja tarkasti niin, ettei testin määrittelyssä ole tulkinnanvaraisia kohtia. [1] Testit kannattaa kohdentaa selkeästi eri alijärjestelmiin ja niiden suunnitteluasteisiin, jotta vältyttäisiin turhalta päällekkäiseltä testaamiselta. Testiohjelman kattava suunnittelu kasvattaa testien kattavuutta ja ehkäisee erilaisten vikaluokkien ylenkatsomista. [2]

Yksi testaussuunnitelman haasteista on erilaisten testitilanteiden rajaton lukumäärä. Eri käyttötilanteita, ympäristöjä ja vikaantumistapoja on lukemattomia, ja testaukseen käytettävät resurssit kuten aika, raha, henkilöstö ja laitteisto ovat rajalliset. Testaussuunnittelun oleellinen tehtävä onkin valita joukko testejä, jotka kattavat mahdollisimman paljon vikoja mahdollisimman vähällä testien määrällä. Tässä käytetään avuksi olettamuksia eri testitapausten päällekkäisyyksistä ja todennäköisimmistä käyttötilanteista, jotka täytyy testata. [2]

Käytännön testaus aloitetaan eri alijärjestelmien ja komponenttien itsenäisen toiminnan testaamisella. Tämän testausvaiheen kohteena ovat järjestelmän matalan tason alijärjestelmät kuten piirikortit ja aliohjelmat ja joissain tapauksissa jopa yksittäiset komponentit. Alijärjestelmän rajapintojen toiminta varmistetaan määrittelyn mukaiseksi ja alijärjestelmän sisäiseen toimintalogiikkaan perehdytään, jotta saadaan selvitettyä alijärjestelmän rakenteellisia vikoja, jotka voivat vaikuttaa toiminnan laatuun ja alijärjestelmän luotettavuuteen eri tilanteissa.

Itsenäisen testauksen aikana testattavan alijärjestelmän rajapintojen herätteet voidaan usein simuloida [1], [4]. Simuloinnin avulla alijärjestelmää on mahdollista käyttää, vaikka rajapintoihin vaikuttavat muut järjestelmän osat eivät olisi valmiita. Simulointimalli on myös todellisia alijärjestelmätoteutuksia yksinkertaisempi ja testaajan kontrolloima kuormitus, mikä varmistaa että löydetyt viat johtuvat testauksen kohteesta

eivätkä muista testiä tukevista komponenteista. Rajapintojen simulointi on helpointa toteuttaa, kun järjestelmä on selkeästi modulaarinen [1], [4].

Integrointivaiheessa testaus jatkuu eri alijärjestelmien keskinäisen toiminnan testaamisella. Alijärjestelmiä yhdistetään keskenään pareittain ja testataan niiden toimintaa niin, että testin ulkopuoliset järjestelmän osat ovat tarpeen mukaan simuloituja. Testausjärjestys riippuu siitä, mitä alijärjestelmiä on toteutettu ja testattu itsenäisesti. Kun eri alijärjestelmäyhdistelmien toimintaa on testattu tarpeeksi, kootaan järjestelmää vaihe vaiheelta valmiimmaksi. Alijärjestelmien integroinnin päämääränä on ratkaista suurimmat rajapintaongelmat ennen kuin koko järjestelmän toimintaa aletaan testata. Kun koko järjestelmä on integroitu, aloitetaan järjestelmätestauksen ensimmäiset vaiheet. [1], [4] Järjestelmätestaus keskittyy järjestelmään kokonaisuutena ja testauksen painopiste on järjestelmän ja sen ympäristön välinen rajapinta. Järjestelmätestauksen menetelmiä käsitellään syvällisemmin luvussa 6.

Järjestelmätestauksen loppuvaiheessa suoritetaan hyväksyntätestaus. Kun järjestelmä on integroitu ja todettu suunnittelu- ja testausorganisaation puolesta toimivaksi, pitää järjestelmälle tehdä joukko testejä, joilla vianetsinnän sijasta osoitetaan järjestelmän toiminta eri tilanteissa. Hyväksyntätestauksen pyrkimyksenä on varmistaa, että järjestelmä täyttää kehittävän organisaation ja käyttäjän näkemykset ja vaatimukset järjestelmän toiminnasta. [1], [4]

Hyväksyntätestaus tehdään lopullista tuotetta vastaavalla järjestelmällä todellisuutta vastaavassa ympäristössä ja testeissä keskitytään todellisiin käyttötilanteisiin [1], [4]. Hyväksyntätestausvaiheessa ei voida kuitenkaan toistaa kaikkia järjestelmän testejä uusimmalla versiolla, joten tuloksissa täytyy luottaa joidenkin ominaisuuksien toimivan hyväksyntävaiheessa, jos ne ovat toimineet aiemmissakin kehitysvaiheissa.

Osa hyväksyntävaiheen vaatimuksista tulee ulkopuolisilta hyväksyntälaitoksilta. Nämä testit täytyy aina toteuttaa, jos kyseisen laitoksen hyväksyntä halutaan. Ulkopuolisen hyväksyntälaitoksen vaatimusten mukainen testaus täytyy suorittaa hyväksyntälaitoksen sertifioimassa laboratoriossa. [1]

Kaikilla testauksen vaiheilla on yhteisiä piirteitä. Alijärjestelmän voi mieltää myös itsenäisenä järjestelmänä, jonka itsenäinen testaus on itse asiassa alijärjestelmän järjestelmätestausta. Testauksen kohde vain kasvaa ja monimutkaistuu alijärjestelmien yhdistyessä. Testauksen kaikissa vaiheissa oleellisia painopisteitä ovat käytetyt testausmenetelmät ja testilaitteisto. Molempien suunnittelussa on omat piirteensä, jotka määrittävät testauksen laatua. Luvussa 4 tutkitaan eri testausvaiheissa käytettyjä testausmenetelmiä ja luvussa 5 testilaitteiston toteutukseen liittyviä periaatteita.

4 Testausmenetelmät

Testausmenetelmät ovat keinoja, joilla testejä suunnitellaan ja toteutetaan. Kun testaussuunnitelmaa tehdessä tiedostaa, minkä tyyppisiä menetelmiä testauksen aikana on mahdollista käyttää, voi suunnittelussa kohdentaa oikeita keinoja oikeisiin tilanteisiin.

Testauksen voi jakaa karkeasti kahteen testaustapaan: Valkoisen laatikon testaaminen (white box testing) ja mustan laatikon testaaminen (black box testing). Valkoisen laatikon testaamisessa testaaja tuntee testauskohteen sisäisen toiminnan ja pystyy käyttämään tätä tietoa hyödyksi testien suunnittelussa. Valkoisen laatikon testeissä yritetään etsiä matalan tason vikoja, jotka ovat luonteeltaan rakenteellisia ja johtuvat komponenteista, ohjelmakoodista, väylistä tai muista alijärjestelmän osista. [1], [4]

Valkoisen laatikon testausmenetelmissä käydään yleensä läpi alijärjestelmän toiminta erittäin tarkasti. Testit vaativat testauksen kohteen tarkkaa tuntemusta ja laajemmissa alijärjestelmissä kaikkien rakenteellisten ominaisuuksien kattava testaaminen vaikeutuu. Toinen valkoisen laatikon testausmenetelmien puute on, että suunnitelluissa testeissä jätetään yleensä täysin huomioimatta järjestelmän toiminnan tarkoitus, kun keskitytään pelkästään sisäisten ratkaisujen toimivuuteen. Näistä syistä laajempien alijärjestelmäkokonaisuuksien toimintaa tutkimaan tarvitaan mustan laatikon testausmenetelmiä täydentämään valkoisen laatikon testausta. [1]

Mustan laatikon testaamisessa ei hyödynnetä tietoa kohteen sisäisestä toiminnasta, vaan testit toteutetaan pelkästään kohteen sisään- ja ulostuloja käyttämällä. Testejä suunnitellessa käytetään apuna tietoa järjestelmän vaatimuksista, käyttötarkoituksesta ja käyttökohteista. [1], [2] Mustan laatikon testauksella etsitään kohteesta toiminnallisia vikoja antamalla sen sisääntuloihin erilaisia herätteitä ja tarkkailemalla niiden aiheuttamia muutoksia ulostulojen tiloissa. Jos ulostulon tila on erilainen kuin annetulla sisääntulolla on suunniteltu, on kohteesta löytynyt vika. Erilaisia mahdollisia vian aiheuttajia ovat suunnittelemattomat käyttötavat, joita ei ole käsitelty järjestelmän toiminnassa, ääriolosuhteet, joilta järjestelmää ei ole suojattu tarpeeksi hyvin sekä käytön ääritilanteet, joissa järjestelmän suorituskyky ei olekaan odotetulla tasolla. Mustan laatikon testauksessa on tärkeää keskittyä keskivertokäytön sijasta nimenomaan käytön ääritilanteisiin, joissa vikaantumiset ovat todennäköisempiä. [4]

Järjestelmän testauksessa valkoisen laatikon testausmenetelmien rooli korostuu toteutuksen varhaisessa vaiheessa, alijärjestelmien ja komponenttien itsenäisessä testauksessa [1], [4]. Valkoisen laatikon menetelmillä tutkitaan yksityiskohtaisesti alijärjestelmän toimintaa, kuten esimerkiksi ohjelmistosuunnittelun puolella muistin- ja prosessoriajan hallinnan kaltaisia ja laitteiston suunnittelussa jännitetasojen, virranlaadun, kytkentälogiikan ja komponenttien toiminnan kaltaisia ominaisuuksia. Mustan laatikon testausmenetelmät tulevat hyödyllisemmäksi järjestelmän toteutuksen ollessa melko kypsällä asteella integrointi- ja hyväksyntävaiheessa, jolloin eri kokonaisuudet toimivat sisäisesti ja niiden rajapintoja voi hyödyntää testaukseen [1]. Molempia testaustapoja on silti mahdollista ja kannattavaa käyttää koko järjestelmän kehityksen ajan. [4] Myös komponenttien ja yksinkertaisten alijärjestelmien rajapinnan toiminta pitää varmistaa mustan laatikon testauksella. Myöhemmissä kehitysvaiheissa vikoja voidaan etsiä mustan laatikon menetelmillä ja vian löytyessä sen syntymisen syitä

tutkitaan tarkemmin valkoisen laatikon menetelmillä. Suurten alijärjestelmäkokonaisuuksien perusteellinen läpikäyminen valkoisen laatikon testausmenetelmillä vie yleensä liian paljon resursseja ollakseen järkevää.

Valkoisen laatikon testausta toteuttaa usein alijärjestelmän suunnittelusta vastaava organisaatio, sillä testaustapa vaatii järjestelmän tuntemusta, joka suunnittelijoilla on valmiina. Mustan laatikon testaamista suunnitteluorganisaation ei sen sijaan ole kannattavaa tehdä, sillä itse suunnittelemaansa alijärjestelmää käyttää todennäköisesti niin kuin on suunnitellut. Tällöin on riskinä jättää tutkimatta käyttötapoja, joita ei itse alun perin ole ajatellut. Mustan laatikon testaus onkin itsenäisen testausorganisaation yleisimmin käyttämä testaustapa. [4] Valtaosa järjestelmätestauksesta toteutetaan mustan laatikon testausmenetelmillä, joten näihin perehdytään tässä työssä syvällisemmin.

Kaikkiin testausvaiheisiin ja -tapoihin sisältyy perusperiaatteita, jotka parantavat testauksen laatua. Näitä toimintatapoja noudattamalla pystytään varmistamaan, että testauksen toteutus on mahdollisimman kattavaa ja testien tuloksia pystyy hyödyntämään tämän hetken lisäksi myös tulevaisuudessa. Testauksen suunnittelun tärkeitä osioita ovat tarvittavien testien valinta, eri testien priorisointi sekä testien oikean toteutuksen varmistaminen. Näitä asioita käsitellään aliluvuissa 4.1 ja 4.2. Molemmissa aliluvuissa keskitytään mustan laatikon testauksen näkökulmaan.

4.1 Mustan laatikon testien valintaperusteita

Ennen kuin yksittäisiä testejä voidaan suunnitella, täytyy päättää toteutettavan testauksen kattavuus ja valita kriittisimmät testipisteet käytettävissä olevien resurssien perusteella. Tarvittavan testauksen suuruusluokka voidaan määritellä arvioimalla testauskohteen kaikkien mahdollisten virheiden määrää eri suunnitteluvaiheissa ja päättää kaikista virheistä osuus, joka testaamalla on realistista löytää. [2], [4]

Virheiden kokonaismäärän arvioimiseksi voidaan käyttää tietoa aikaisemmin suunniteltujen järjestelmien virheiden määrästä sekä verrata järjestelmää kirjallisuudesta löytyviin teollisuuden keskiarvoihin, jos järjestelmän monimutkaisuus vastaa löydettyjä arvoja. Testauksen tarvetta voidaan myös arvioida uudelleen testauksen alkuvaiheissa havaitun virhetiheyden perusteella. Jos jostakin järjestelmän osasta ei löydy odotettua määrää vikoja, voidaan testauksen painopistettä siirtää osioihin, joissa virheitä löytyy alusta alkaen odotettua enemmän. Ongelmalliseksi havaittu järjestelmän osa sisältää todennäköisesti paljon myös aikaisemmin havaitsemattomia vikoja. Lisäksi, jos suunniteltu järjestelmä vaikuttaa epäilyttävän virheettömältä, voidaan käyttää ulkopuolista tahoa selvittämään onko järjestelmä todellisuudessa virheetön vai onko testausjärjestelyissä puutteita. [2]

Kun tarvittava testauksen taso on päätetty, aletaan priorisoida ja valita testitilanteita tarkemmin. Kaikki yksittäiset alijärjestelmät kannattaa testata valkoisen laatikon menetelmillä alijärjestelmän sisäisten osioiden toiminnan laadun ja tehokkuuden ja mustan laatikon menetelmillä alijärjestelmän rajapintojen oikean toiminnan varmistamiseksi. [2]

Jos kyseessä on järjestelmä tai järjestelmän osa, jolla on useita mahdollisia yhteensopivia laitteisto- ja ohjelmistokonfiguraatioita, voi olla vaikeaa löytää kohtuullinen määrä tarpeellisia yhdistelmiä joiden yhteensopivuus testataan. Tällöin kannattaa priorisoida asiakkaiden yleisimmin käyttämiä vaihtoehtoja, koska näissä huomaamatta jääneillä suunnitteluvirheillä on todennäköisimmin seurauksia valmiin järjestelmän käytössä. Mitä kriittisempää konfiguraation toimiminen on järjestelmän toiminnalle, sitä useammalla eri konfiguroinnilla sen yhteensopivuus pitää testata. [4]

Tutkiva testaaminen, jossa kokenut testaaja etsii testauksen aikana erilaisia virhetilanteita suorittamalla lennosta kokeita järjestelmälle, voi parhaimmillaan olla halpa ja nopea tapa löytää odottamattomia vikoja. Testauksen onnistuminen riippuu kuitenkin täysin yksilön kyvyistä ja intuitiosta, eikä sen takia ole ainoana menetelmänä testausstrategian kannalta kestävä ratkaisu. [2], [4] Testausstrategiassa voi olla hyödyllistä yhdistää tutkivaa testausta suunniteltuihin testeihin. Suunniteltujen testien avulla verrataan järjestelmän toimintaa sen määrittelyyn ja varmistetaan, että se toimii odotetusti, ja tutkivalla testauksella etsitään odottamattomia tilanteita ja ongelmia, joita järjestelmän määrittelyvaiheessa ei ole huomioitu. [1]

Mustan laatikon testaaminen perustuu rajapintojen käyttöön. Testattavien tilanteiden määrittämiseksi pitää selvittää testauskohteen rajapintojen tilat, jotka järjestelmä käsittelee eri tavalla ja testata näistä jokainen. Tätä kutsutaan ekvivalenssiluokkiin jaotteluksi (equivalence partitioning). Ekvivalenssiluokkia ovat sisääntulon tilat, jotka käsitellään järjestelmässä keskenään samanlaisesti. Esimerkiksi sisääntulolle, joka tekee asiaa A lukuarvoilla 0 – 10 ja asiaa B lukuarvoilla 11 – 20 ja jolle muut lukuarvot ovat virheellisiä syötteitä, voidaan määritellä neljä ekvivalenssiluokkaa. Arvot, jotka ovat pienempiä kuin 0, arvot väliltä 0 – 10, arvot väliltä 11 – 20 ja arvot, jotka ovat suurempia kuin 20. Edellä mainitut neljä luokkaa käsitellään testattavassa järjestelmässä eri tavoilla, joten jokainen luokka tulisi sisällyttää erikseen testiohjelmaan. [1], [2]

Osa ekvivalenssiluokista on sallittuja tiloja ja osa virheellisiä, ja niiden vaatimukset ovat näin keskenään erilaisia. Virheellinen tila tulee käsitellä sen vakavuuden vaatimalla tavalla sallituksi; Turvakriittisen toiminnon tapauksessa tämä tarkoittaisi todennäköisesti toiminnon hätäpysäytystä, kun puolestaan toissijaisen toiminnon tapauksessa järjestelmä voi rajoittaa itsenäisesti virheellisen arvon sallittuun tilaan tai ilmoittaa asiasta käyttäjälle. Sallitun tilan toiminta pitää puolestaan testata oikeanlaiseksi kaikissa sille odotettavissa olevissa tilanteissa: eri ympäristön olosuhteissa, kuormituksissa ja alkuasetelmissa. Mahdollisten virhetilojen määrittely on tehtävä erityisen huolellisesti, vaikka niitä ei selkeästi erottuisikaan [2]. Käsittelemättömät virhetilat saattavat aiheuttaa hankalia ja jopa vaarallisia toimintoja valmiissa järjestelmässä.

Eri ekvivalenssiluokkien sisältä valitut testipisteet kannattaa valita luokan rajojen läheisyydestä sen ollessa mahdollista. Sallitun ja ei-sallitun tilan, kuten jonkin arvon sallitun maksimin, rajapinnalla tapahtuu yleensä paljon virhetilanteita. [1], [2] Kun järjestelmä toimii rajan läheisyydessä olevalla sisääntulon arvolla, voidaan olettaa, että turvallisemmin joukon sisällä olevat arvot toimivat samalla tavalla. Sisääntulojen rajatapausten testaamisen lisäksi pitää huomioida, että ulostulojen rajatapaukset saattavat ilmetä eri arvoilla kuin sisääntulon rajatapaus [2]. Myös nämä tulisi testata.

Testien määrän vähentämiseksi voidaan määrittää rajapinnat, komponentit ja ohjelman osat, jotka ovat keskenään rinnasteisia. Keskenään rinnasteiset järjestelmän osat ovat ominaisuuksiltaan ja toiminnaltaan täysin vastaavia mekaanisesti, ohjelmistoltaan ja toimintalogiikaltaan. Rinnasteisista rajapinnoista voidaan testata pelkästään yhtä ja olettaa sen testien tuloksesta, onko muissa rinnasteisissa rajapinnoissa vikoja. [2] Tällaisia rinnasteisia osia ovat esimerkiksi keskenään identtiset sisään- ja ulostulot.

Etsittäessä testattavia toimintoja täytyy selvittää järjestelmällisesti kaikki mahdolliset tavat käyttää testauksen kohdetta. Monimutkaisissa järjestelmissä eri sisään- ja ulostulojen toimintatavat voidaan määritellä järjestelmällisesti syy – seuraus -taulukoita käyttämällä. Testinsuunnittelun perustana käytettävistä dokumenteista, kuten käyttäjädokumentaatiosta tai järjestelmän määrittelystä, taulukoidaan kaikki testattavan kohteen sisään- ja ulostulot ja merkitään niiden väliset riippuvuudet. Sisään- ja ulostulojen keskinäiset riippuvuudet voidaan määritellä boolean-logiikan avulla. [1], [2] Jokaisen portin jokaiselle tilalle merkitään yksilöllinen tunnus kuten *S1T1* tai U2T2 (Sisääntulo 1:Tila 1, Ulostulo 2:Tila 2). Toisiinsa vaikuttavien porttien suhteet voidaan tämän jälkeen taulukoida esimerkiksi merkitsemällä *S1T1 JA S2T3 => U1T2*. Kaikkien ehtojen kombinaatioiden läpikäyminen on työläs prosessi, mutta jos järjestelmän toiminnan testaaminen halutaan tehdä mahdollisimman kattavasti, on syy – seuraus -taulukointi perusteellinen apuväline.

Yllä mainituilla menetelmillä saadaan rajattua testaustapauksia hallittavissa olevaksi joukoksi. Testattavien ominaisuuksien valinnan jälkeen suunnitellaan jokainen testi erikseen, jotta ne voidaan toteuttaa hallitusti. Tätä käsitellään seuraavassa aliluvussa.

4.2 Testien toteutus

Testin toteutus koostuu kolmesta osasta: testin määrittely, suorittaminen ja raportointi. Kaikki kolme osiota ovat tärkeitä ja jokainen täytyy suunnitella huolellisesti, jotta testauksen laatu on mahdollisimman hyvä.

Testiä määritellessä täytyy päättää käytettävät sisääntulot ja niihin asetettavat arvot, tarkasteltavat ulostulot sekä ulostulojen odotusarvot testin aikana. Myös testin olosuhteet kuten esimerkiksi ympäristön lämpötila, kosteus ja häiriöisyys on määriteltävä. Jos testille määritellään erityinen olosuhde, pitää testivaatimuksissa ohjeistaa, miten kyseinen olosuhde saavutetaan. Tämän saavuttamiseksi testin määrittelyssä ilmoitetaan myös testissä käytettävä laitteisto. Myös selkeä läpäisyehto ja vaatimukset, joihin saatuja tuloksia verrataan, pitää olla selvillä ennen testin aloittamista. Jokaisella tehdyllä testillä pitää olla jokin analysoitava tulos, jota järjestelmän suunnittelussa voidaan hyödyntää. Testejä määritellessä pitää myös olla selvillä, mitä ominaisuuksia kyseinen testi kattaa, jolloin sen rooli testaussuunnitelmassa on selkeä. [1], [2] Jos itse testaustilanteessa on todennäköistä tai jopa haluttua, että testauksen kohde tuhoutuu testin aikana, kannattaa se toteuttaa testausohjelmassa viimeisenä ja kiinnittää erityistä huomiota testiympäristön turvallisuuteen [4]. Näin testattavan laitteiston hajoaminen ei aiheuta ylimääräistä vahinkoa, ja korjauksiin vaadittu aika ei hidasta muiden testien toteutusta.

Jos testi on määritelty hyvin, on sen suorittaminen kokeneelle testaajalle melko suoraviivaista. Testin aikana pitää huolehtia tarvittavien välineiden hankinnasta, kalibroinnin varmentamisesta ja tarvittaessa kalibroinnista sekä oikeasta käytöstä. Jos testauksen määrittelyssä on epäselvyyksiä, jää testaajan tehtäväksi selvittää epäselvyydet tutkimalla testitilannetta kokemuksensa perusteella ja kysymällä testin suunnitelleelta taholta tarkennusta tarvittaessa. Jos testauksen aikana huomataan puutteita testien kattavuudessa joillain osa-alueilla, voi olla tarvittavaa myös täydentää testausta lennosta. Näidenkin testien määrittely, toteutus ja tulokset tulisi kuitenkin raportoida samalla tavalla kuin etukäteen suunnitellut testit, jotta ne ovat tarvittaessa toistettavissa ja mahdollisesti lisättävissä järjestelmän testaussuunnitelmaan vakituisemmin [2].

Kun testauksen aikana löytyy vikoja, korjataan niitä tilanteen mukaan joko samaan testilaitteistoon tai seuraaviin versioihin. Tämän jälkeen vähintään viallista ominaisuutta koskevat testit toistetaan, jotta varmennutaan korjauksen onnistumisesta. Järjestelmän versiomuutosten välistä uudelleentestausta kutsutaan regressiotestaukseksi. Ideaalisesti regressiotestauksessa uusi versio testattaisiin täydellisesti uudestaan, koska erityisesti ei-modulaarisissa järjestelmissä on muutoksilla mahdollista aiheuttaa odottamattomia seurauksia muihin järjestelmän toimintoihin kuin on tarkoitus. Kaikkien testien uusiminen kaikissa vaiheissa veisi kuitenkin valtavasti resursseja, joten testausorganisaation täytyy tehdä kompromissi testeistä, joita eri versioille toteutetaan. [4] Regressiotestausta helpottaa, jos jokaisen muutoksen jälkeen selvitetään, mihin muutos on voinut vaikuttaa ja mitkä testit täytyy tämän vaikutuksen takia uusia muutoksen jälkeen. [1]

Korjausten lisäksi löydettyjen vikojen syitä on hyödyllistä analysoida tarkemmin, jotta samoja virheitä ei toistettaisi tulevissa suunnitteluprojekteissa. Mahdollisia selvitettäviä asioita ovat vian syntypaikka suunnitteluketjussa, eli syntyikö vika määrittelyn, suunnittelun vai toteutuksen aikana, miksi virhe tapahtui, eli onko kirjoitetuissa määritelmissä ollut virhe, onko kyseessä ollut huolimattomuus tai koulutuksen puute, kuinka virhe voidaan seuraavalla kerralla estää, miksi vikaa ei havaittu aiemmin ja kuinka se olisi voitu havaita aiemmin. Vian analysointi saattaa perusteellisesti tehtynä olla raskas ja kallis prosessi. Se voi kuitenkin olla erittäin hyödyllistä seuraaville projekteille ja säästää niissä turhia kustannuksia. [2] Virheiden tarkempi analysointi voi olla kannattavaa erityisesti kriittisissä toiminnoissa, koska niistä aiheutuvat ongelmat ovat erityisen kalliita ja jopa vaarallisia.

Testi täytyy raportoida, jotta sen suorittamisesta on hyötyä. Raportoinnin tarkoituksena on ilmoittaa testin tulokset muodossa, jossa niitä voidaan verrata testin vaatimuksiin ja lisäksi yhdessä testinmäärittelyn kanssa arkistoida testin toteutustapa niin, että sen voi tarvittaessa toistaa myöhemmin. Raportissa pitää tulosten ja niiden odotusarvojen lisäksi ilmoittaa käytetty testilaitteisto ja ympäristön olosuhteet [1], [4]. Raportti ja testimäärittely lukemalla pitäisi kenen tahansa asiantuntijan kyetä toistamaan testi täysin identtisesti alkuperäiseen nähden ja verrata saamiaan tuloksia aikaisempaan. Tämä on tärkeää regressiotestauksen ja myöhemmän testitulosten varmistamisen kannalta.

5 Testilaitteisto

Testilaitteisto on suunniteltava sellaiseksi, että se täyttää toteutettavalle testille määritellyt ympäristön ja kuormituksen vaatimukset, kerää ja tallentaa tietoa halutulla tavalla eikä vaikuta itse mittaustuloksiin esimerkiksi mittauspiirin kytkeytymisen takia. Testijärjestelyn täytyy olla hallittavissa niin, että sillä tehdyt testit voidaan myöhemmin toistaa täsmälleen samoissa olosuhteissa. [1], [4] Testilaitteiston luotettavuutta pystytään varmentamaan injektoimalla testattavaan järjestelmään tunnettuja vikoja, jotka testilaitteiston pitäisi havaita. Joskus voidaan käyttää vertailukohtana myös testattavan järjestelmän sisäistä diagnostiikkaa, jos se tarkkailee samoja arvoja kuin testilaitteisto. Järjestelmän diagnostiikan toiminta täytyy tätä ennen kuitenkin testata ulkoisilla mittauksilla, jotta tiedetään millä tarkkuudella siihen voi luottaa. [1]

Testilaitteistolle hyviä ominaisuuksia ovat muunneltavuus ja laajennettavuus, joiden ollessa kunnossa testilaitteistoa pystytään käyttämään tulevissa, myös alun perin ennakoimattomissa tilanteissa. Testilaitteistosta kannattaa kehittää myös mahdollisimman helppokäyttöinen, jotta inhimillisen virheen mahdollisuus on pieni. [1], [4] Myös testilaitteiston dokumentointi on tärkeää. Laitteistolle pitää olla selkeät käyttöohjeet ja toiminnan kuvaus [4]. Lähtökohtaisesti testilaitteistosta kannattaa suunnitella selkeäkäyttöinen ja sen pitäisi olla päivitettävissä niin, ettei koko testijärjestelyä tarvitse suunnitella uudestaan muutosten takia. Taloudellisista syistä kannattaa testilaitteisto suunnitella myös sellaiseksi, että sillä pystyy toteuttamaan mahdollisimman paljon keskenään samankaltaisia testejä [1]. Näin varmistetaan testijärjestelyn suunnittelun kustannuksilla saatava mahdollisimman suuri hyöty. Testilaitteiston toiminnan lisäksi kannattaa kiinnittää huomiota myös sen huollettavuuteen. Jokainen testilaitteisto vaatii ylläpitoa, jotta se toimisi mahdollisimman pitkään [1].

Testiympäristön luominen erityisesti laitteiston testeille on haastavaa [1]. Halutut ympäristöt saattavat olla esimerkiksi todella kylmiä tai kuumia, erittäin kosteita, likaisia, pölyisiä tai kemiallisesti rasittavia. Muutenkin aitoja vastaavien käyttötilanteiden luominen valmiille alijärjestelmille voi olla vaikeaa erityisesti testauksen alkuvaiheessa [1]. Järjestelmän puuttuvia osia ja ympäristöä voi simuloida, mutta epäideaalisuudet huomioivissakin simulointimalleissa ei pystytä mallintamaan täydellisesti todellisia käyttötilanteita ja ympäristön vaikutusta.

Testattavaa järjestelmää rasittavia ääriolosuhteita luodessa täytyy varmistaa, ettei testiympäristö rasita testin kohteen lisäksi testilaitteistoa [4]. Tämä voidaan ehkäistä esimerkiksi eristämällä testattava laite kaappiin, joka toteuttaa tarvittavan ympäristön. Tällaisia ovat muun muassa erilaiset kosteutta ja lämpötilaa säätävät sääkaapit. Näin pelkästään mittausanturit ja -piuhat ovat altistettuina kuormittavalle ympäristölle. Jos testilaitteisto on pakko altistaa samalle ympäristölle kuin testattava laite, pitää se suunnitella niin, ettei ympäristö vaikuta testilaitteiston antamiin tuloksiin.

6 Järjestelmätestaus

Tässä luvussa käsitellään tarkemmin järjestelmätestausta, jonka kehittäminen on tämän diplomityön tutkimuskohde. Järjestelmätestaukseen liittyvät samat periaatteet kuin testaukseen yleisestikin. Tämän lisäksi siinä on kuitenkin erityispiirteitä. On myös oleellista selvittää, mitä testaustapoja järjestelmätestausvaiheessa painotetaan.

Järjestelmätestausvaihe alkaa, kun kaikki kriittiset alijärjestelmät on integroitu ja järjestelmää voidaan käyttää kokonaisuutena. Testausvaiheiden vastuualueiden rajat ovat häilyviä, mutta tuotekehityksen järjestelmätestausvaiheen voidaan katsoa alkavan integrointivaiheen loppupäässä kokonaisen järjestelmän integrointitestauksena ja jatkuvan hyväksyntätesteihin sekä ensimmäisten tuotannon pilottijärjestelmien testaukseen. Järjestelmätestaus tehdään usein käyttäjän näkökulmasta, ja tämän takia siinä painotetaan suunnitellun järjestelmän toiminnan testaamista odotetuissa käyttötilanteissa ja -ympäristöissä [4]. Lisäksi testataan järjestelmän kriittisiä ominaisuuksia kuten järjestelmän luotettavuutta ja erilaisten vika- ja ääritilanteiden hallintaa. [1] Järjestelmätestauksen aikana ei pelkästään toisteta aikaisempien vaiheiden testejä uudestaan koko järjestelmälle, vaan tehdään uusia testejä, jotka vertaavat järjestelmän toimintaa sille suunnittelun alkuvaiheessa päätettyihin määritelmiin ja vaatimuksiin. Tätä varten suunnittelun alussa on pitänyt määritellä järjestelmän toiminnan tavoitteet. [2]

Monia järjestelmätestausvaiheen testausmenetelmistä käytetään myös alijärjestelmien testauksessa, mikä on luontevaa, koska jokainen alijärjestelmä on myös itsessään järjestelmä, jolla on omat rajapintansa ja jonka voi testata oman ympäristönsä suhteen kokonaisuutena. Kaikkien alijärjestelmien itsenäinen toiminta ei kuitenkaan osoita vielä järjestelmän toimintaa, minkä takia järjestelmätestausvaihetta tarvitaan.

Järjestelmätestaus on laaja kokonaisuus, johon sisältyy paljon teknologiaan ja käyttökokemukseen liittyviä testikokonaisuuksia. Tässä työssä keskitytään seuraaviin teknologiakeskeisiin testausmenetelmiin: toiminnallinen testaus, suorituskykytestaus, rasitustestaus, sähkömagneettisen yhteensopivuuden testaus ja luotettavuustestaus. Näiden lisäksi järjestelmätestausvaiheen tärkeitä osioita ovat muun muassa käytettävyys, järjestelmän tietoturva ja tuotanto- ja logistiikkaprosessi. Kyseiset seikat ovat oleellinen osa järjestelmän lopullista toteutusta. Nämä testauksen osiot ovat kuitenkin itsenäisiä, erikoistumista vaativia kokonaisuuksia. Tämän takia kyseiset osiot on jätetty työn ulkopuolelle.

6.1 Toiminnallinen testaus

Toiminnallisessa testauksessa keskitytään järjestelmätestivaiheessa kokonaisen järjestelmän ja ympäristön väliseen rajapintaan. Testeissä pitää varmistaa, että kaikki suunnitellut toiminnot on toteutettu ja käyttäjän tehtävissä [2], [4]. Järjestelmätestien suunnittelun pohjana voidaan käyttää käyttäjädokumentaatiota, jonka mukaisesti järjestelmän tulisi eri tilanteissa toimia. Samalla pystytään tarkistamaan käyttäjädokumentaation laatu, ja voidaan löytää myös sitä kautta puutteita ja poikkeavuuksia järjestelmän alkuperäisestä tavoitteesta. [2]

Yksi toiminnallisten testien alalaji on viankäsittelyn testaus. Järjestelmälle on määritelty erilaisten vikatilanteiden kriittisyys, hallinta ja palautumistapa. Nämä ominaisuudet pitää testata, jotta varmistutaan järjestelmän oikeasta toiminnasta kaikissa tilanteissa. Erityisesti turvakriittisissä ja mahdollisesti vaarallisissa järjestelmissä eri vikatilanteiden hallinta on tärkeää testata. [1], [2]

Viankäsittelytestien näkökulma vaihtelee viasta riippuen. Osan vioista pitää katkaista järjestelmän toiminta mahdollisimman nopeasti, osan vioista pitää aiheuttaa hallittu pysäytys, kun puolestaan jotkin viat kuten pieni määrä korruptoitunutta dataa eivät saa vaikuttaa järjestelmän toimintaan lainkaan. Viankäsittelytestit toteutetaan injektoimalla järjestelmään hallitusti tunnettuja vikoja. Tämä tapahtuu esimerkiksi avaamalla tai oikosulkemalla virtapiirejä laitteistosta tai syöttämällä ohjelmistoon viallista dataa. Tämän jälkeen tarkkaillaan järjestelmän reaktiota vikaan. Jos viankäsittely on järjestelmässä automaattista, tarkastetaan palautuuko järjestelmä viasta oikein halutussa ajassa ja antaako se viasta asianmukaista tietoa käyttäjälle. Jos viasta palautuminen on manuaalista, yrittää testaaja vian injektoimisen jälkeen palauttaa järjestelmän toimintaan ja huomioi samalla, palaako järjestelmä toimintakuntoiseksi suunnitellulla tavalla. Lisäksi tilanteissa, joissa järjestelmä käsittelee tietoa, tarkastellaan onko se säilynyt vian jälkeen ehjänä. [1]

6.2 Suorituskykytestaus

Suorituskykytestauksessa järjestelmää kuormitetaan sille suunnitelluilla kuormilla ja tarkkaillaan toimiiko järjestelmä stabiilisti ja ongelmitta [1], [4]. Siinä missä toiminnallisella testauksella tarkistetaan toimiiko järjestelmä oikein, suorituskykytestauksella testataan toimiiko se tarpeeksi hyvin. Testaamalla etsitään suorituskyvyn pullonkauloja, jotka rajoittavat järjestelmän toimintakykyä kuormittaessa. Suorituskykytestausta tehdään myös alijärjestelmille, mutta järjestelmätestausvaiheessa kun testausorganisaatiolla on käytössään kokonainen toimintakuntoinen järjestelmä, voidaan sen suorituskykyä mitata yksiselitteisesti verrattuna järjestelmälle annettuihin tavoitteisiin. [1]

Suorituskykytestausta on tärkeää tehdä todellisissa käyttöympäristöissä, jotta testatun järjestelmän käytön aikana ei havaita suorituskyvyn olevan riittämätön joissakin olosuhteissa. Jos järjestelmän suorituskykyä ei kehitysvaiheessa testata riittävästi ja riittävän aidoissa olosuhteissa, voi se todellisessa tilanteessa olla liian hidas tai epäluotettava ja jopa käyttökelvoton ilman parannuksia. [1]

Suorituskykytestauksessa testattavaa järjestelmää käytetään sen nimellisessä käyttöpisteessä ja tarkkaillaan sen toimintaa. Kuormitusta muutetaan järjestelmän määrittelyn mukaisesti niin, että erilaisia kuormituksia lisätään yksi kerrallaan testijärjestelyyn ja tarkastellaan niiden vaikutusta toimintaan. Kuormituksia, joille järjestelmä altistetaan, ovat esimerkiksi eri ympäristön olosuhteet kuten kosteus ja lämpö, suuremmat datamäärät tai suurempi sisään- ja ulostuloliitäntöjen määrä ja vaihtelevat tai erityyppiset muut kuormat. Testin aikana tarkkaillaan, täyttävätkö järjestelmän suunnitteluparametrit kaikissa tilanteissa niille asetetut vaatimukset. Tarkkailtavia ominaisuuksia ovat esimerkiksi tehtävien vasteajat, kapasiteetti lisäoptioille, ulkoisille ohjauksille ja väyläliitännöille sekä järjestelmän resurssienkäyttö, toiminnan stabiilius ja tarkkuus. [1]

6.3 Rasitustestaus

Rasitustestaus on samankaltaista kuin suorituskykytestaus siinä määrin, että siinä kuormitetaan järjestelmää ja tarkkaillaan sen toimintaa kuormituksen alla. Erona rasitustestissä on kuitenkin se, että siinä järjestelmää kuormitetaan selvästi yli sen normaalien toimintaolosuhteiden – jopa niin, että järjestelmä voidaan yrittää hajottaa kuormituksella. [1], [2]

Rasitustestauksen tavoitteena on varmistaa, että järjestelmä kestää hetkellisen ylikuormituksen ja toimintakuvauksensa mukaan joko jatkaa toimintaa tai pysäyttää toiminnan ja palautuu toimintakykyiseksi kuorman poistuttua. Ankarimmissa rasitustesteissä pyritään varmistamaan, että järjestelmä hajoaa vian sattuessa hallitusti. [1]

Tietojärjestelmille rasitustestausta voidaan toteuttaa epärealistisen suurilla datamäärillä, prosessorin ja muistin suurilla kuormilla ja paljolla väyläliikenteellä [1], [2]. Laitteiston testaukseen voidaan käyttää esimerkiksi HALT-testausta (Highly Accelerated Life Test) ja komponenttien hajoamistestejä [1].

HALT-testauksessa järjestelmää käytetään eri ympäristön rasitusten alaisena. Yleisiä ympäristökuormituksia ovat lämpö, kosteus ja tärinä [1], [4], [5]. Näistä kuitenkin kosteuden vaikutukset järjestelmään syntyvät pitkän ajan kuluessa, minkä takia se saatetaan jättää pois testauksesta tulosten nopeuttamiseksi [5]. HALT-testissä lisätään kuormitusta vaihe vaiheelta niin pitkään, että jokin järjestelmän osa vikaantuu. Vikaantumisen syy selvitetään ja korjataan ja laite palautetaan toimintakuntoiseksi. Kuormitusta lisätään vaiheittain niin pitkään, kunnes järjestelmän parantaminen ei enää ole taloudellisesti kannattavaa. Ympäristön kuormituksia testataan yhtä kerrallaan, ja kun kaikki halutut olosuhteet on testattu, voidaan niitä myös yhdistää keskenään. [1]. Testiin sisällytetään yleensä myös nopeita olosuhdemuutoksia, kuten lämpötilan nopea muutos, jolloin muutosnopeus aiheuttaa järjestelmälle rasitusta [5]. HALT-testauksella etsitään järjestelmän tuhoutumisen rajoja ja nostetaan järjestelmän kestoa parantamalla heikoimpia komponentteja. [1], [5]

Järjestelmän tuhoavissa testeissä saatetaan jokin järjestelmän osa hajoamaan tarkoituksellisesti ja seurataan kuinka hallittu hajoaminen on. Tuhoavat testit kannattaa tehdä testauksen loppuvaiheissa, kun prototyyppilaitteen hajoaminen ei pysäytä muuta testausta, vaan uusia laitteita on saatavilla helposti [4]. Tuhoavien testien ongelma on se, että niitä ei ole taloudellista toistaa kovin useita kertoja, ja niitä testataan tämän takia melko pienellä otannalla. Tästä saattaa syntyä vääristymä siitä, että kaikki laitteet tuhoutuvat kaikissa tilanteissa samalla tavalla, mikä ei välttämättä ole totta. Tämän ongelman ratkaisemiseksi on joissakin järjestelmäkehitysprojekteissa simuloitu tuhoavia testejä, jotta saadaan jonkinlaisia arvioita seurauksista. Fyysisen laitteen tuhoutuminen on kuitenkin haastavaa mallintaa realistisesti. [1]

6.4 EMC-testaus

Nykyaikana ympäristössä on perinteisten kuormittavien olosuhteiden lisäksi paljon sähkömagneettisia häiriöitä. Erilaisia sähkölaitteita on lähes kaikissa käyttöympäristöissä. Myös suunnitellun sähköisen järjestelmän osat itsessään aiheuttavat sähkömagneettisia häiriöitä ympäröiville laitteille ja muille saman järjestelmän osille. Lisäksi luonnonilmiöt, kuten salama, aiheuttavat häiriöitä sähköjärjestelmiin. Sähkömagneettiset häiriöt aiheuttavat kohteessaan vikavirtoja ja -jännitteitä, joista voi

seurata kriittisiäkin virhetoimintoja. Tämän takia sähkömagneettiselle yhteensopivuudelle on määritelty lakisääteiset rajat, ja järjestelmän yhteensopivuus täytyy tuotekehitysvaiheessa testata sähkömagneettisen yhteensopivuuden testeillä eli EMC-testeillä (Electromagnetic Compatibility). [6] Sähkömagneettinen yhteensopivuus täytyy testata sekä järjestelmän aiheuttamien häiriöiden, että häiriösietoisuuden kannalta [1], [6], [7]. Sähkömagneettinen yhteensopivuus täytyy testata kaikilla järjestelmän osilla, kuten piirikorteilla ja signaaliteillä erikseen sekä järjestelmällä kokonaisuutena [6].

Sähkömagneettiset häiriöt kulkevat joko johtimia pitkin tai säteilemällä [6], [7]. Tästä syystä sähkömagneettisen yhteensopivuuden testaus jaetaan johtuvien häiriöiden ja säteilevien häiriöiden testaamiseen. Johtuvia häiriöitä testataan syöttämällä laitteeseen sen syöttökaapeleiden kautta hallittuja virta- ja jännitehäiriöitä ja tutkimalla laitteen toimintaa. Laitteen aiheuttamat häiriöt voidaan puolestaan mitata esimerkiksi spektrianalysaattorilla vertaamalla syöttöverkon sähkönlaatua ilman laitetta siihen, kun laite on kytketty verkkoon. Testeissä voidaan käyttää erikseen rakennettua mittausverkkoa, jonka impedanssi on tunnettu. Säteilevät häiriöt pitää testata sähkömagneettisesti heijastamattomassa kammiossa, jossa järjestelmää häiritään ja sen aiheuttamia häiriöitä mitataan antenneilla. [6]

6.5 Luotettavuustestaus

Luotettavuustestauksen avulla pyritään selvittämään järjestelmän vikaantumistiheyttä sekä herkimmin vikaantuvia komponentteja, jotta järjestelmän luotettavuutta saataisiin nostettua sen eliniän aikana [1], [5]. Testattavalle järjestelmälle määritellään normaali käyttöprofiili ja ajetaan tätä profiilia niin pitkään, että merkittävä joukko erilaisia vikoja ja niiden tiheyksiä saadaan tilastoitua. Tilastollisesti merkittävä testattavien laitteiden ja vikaantumisten määrä on oleellista, sillä luotettavuustestien tuloksista pitäisi kyetä tilastollisilla menetelmillä määrittelemään kohteen luotettavuuteen vaikuttavia tekijöitä ja kehittämään niitä. [1], [4] Jos luotettavuuteen liittyviä ratkaisuja tehdään liian pienellä testien määrällä, on vaarana tehdä suunnittelutoimenpiteitä sattumaan perustuvien testitulosten varassa.

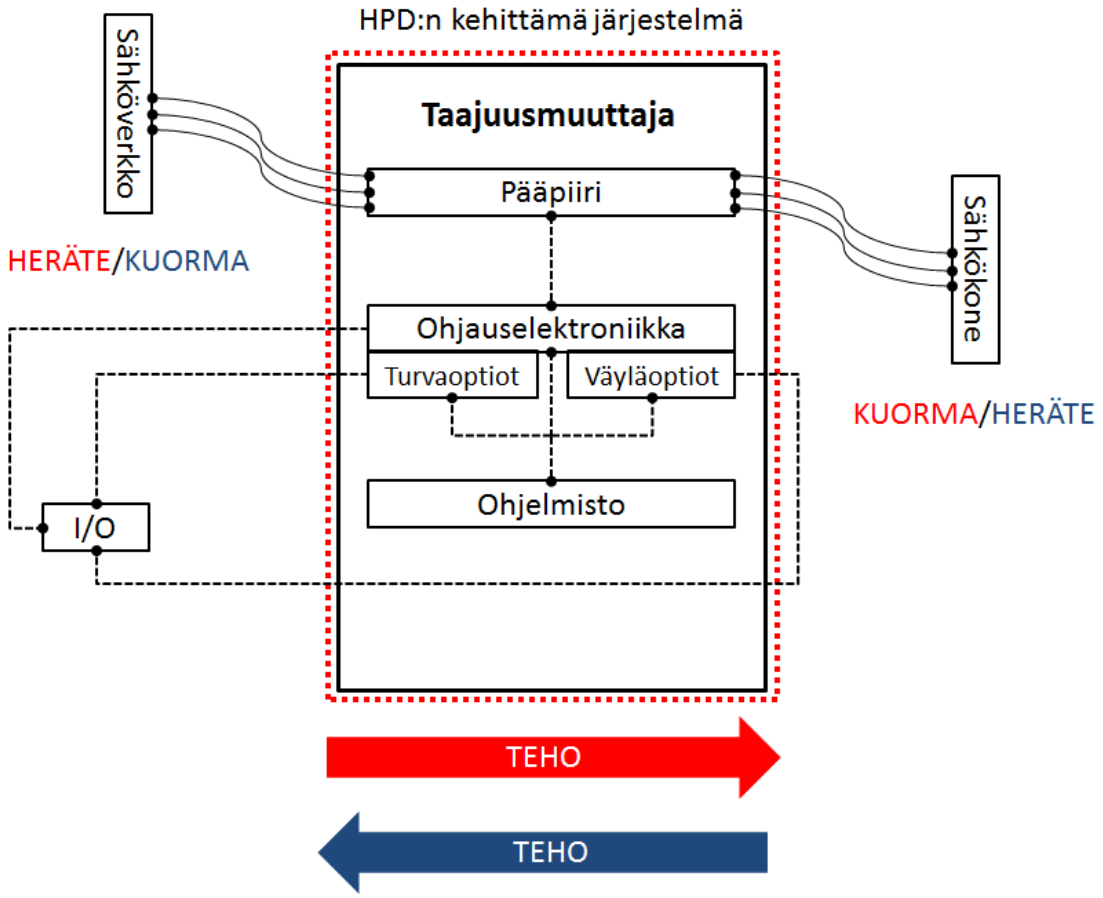
Eräs yleisesti käytetty luotettavuustestausmenetelmä on kiihdytetty elinikätestaus eli ALT-testaus (Accelerated Life Testing). ALT-testeissä järjestelmää kuormitetaan suunniteltuun käyttöön verrattuna kiihdytetysti esimerkiksi käyttämällä järjestelmää normaalia korkeammassa ympäristön lämpötilassa tai suuremmalla teholla. Näin järjestelmä saadaan vanhenemaan normaalikäyttöä nopeammin, ja tämän seurauksena sen ikääntymiseen liittyvät viat ilmenevät lyhemmässä ajassa ja vikaantumistiheys pystytään määrittämään nopeammin kuin kiihdyttämättömällä testauksella. ALT-testien kuormituksille määritellään kiihdytyskerroin, jonka avulla testissä havaitun vikaantumistiheyden perusteella pystytään laskemaan normaalikäytön vikaantumistiheys järjestelmän eri komponenteille. ALT-testauksen onnistumiseksi testaajan pitää tuntea kiihdytetyt kuormitukset ja niiden vaikutus järjestelmään tarkasti. Kiihdytettyä kuormitusta suunniteltaessa on riskinä muuttaa järjestelmän vikaantumistapoja niin, että ne eivät enää vastaa normaalia käyttöä. Tämän takia testiä määritellessä täytyy kiihdytetyt kuormitustekijät ja niiden kiihdytyskerroin valita sellaisiksi, että järjestelmä vanhenee ja vikaantuu samoilla tavoin kuin normaalikäytössäkin – ainoastaan nopeammin. [5]

7 Taajuusmuuttaja järjestelmänä

Taajuusmuuttaja on laite, joka muuttaa syöttöverkon jännitteen tason ja taajuuden halutuksi, ja syöttää muutetun sähkön ulostuloonsa. Tämän johdosta taajuusmuuttajalla pystytään esimerkiksi ohjaamaan moottorin akselin pyörimisnopeutta ja vääntömomenttia. Taajuusmuuttajia käytetään pumppujen ja puhaltimien ohjaamiseen, sähkövoimalaitoksen kytkemiseen sähköverkkoon tai teollisuuslaitosten konelinjastoissa, joissa eri akselien pyörimisnopeutta ja momenttia pitää säätää. Tällainen on esimerkiksi paperikone. Taajuusmuuttajan etuja verrattuna mekaaniseen prosessin ohjaukseen vaihteistolla tai säätöventtiileillä on säädön tarkkuus ja energiansäästö. [7]

ABB valmistaa Helsingin tehtaassaan monia erilaisia taajuusmuuttajia. Pienitehoiset taajuusmuuttajat on rakennettu pääasiassa kokonaisuutena seinälle ripustettavaksi, kun puolestaan HPD:n kehittämät suuritehoiset taajuusmuuttajat on jaettu syöttö- ja lähtöpuolen moduuleihin, jotka asennetaan kaappeihin. Tehon kasvaessa sekä syötön että lähdön kuorma voidaan jakaa usealle keskenään rinnan kytketylle moduulille. Erityinen taajuusmuuttajatyyppi on niin kutsuttu Multidrive, jossa yksi syöttöyksikkö syöttää useaa lähtömoduulia, jotka ohjaavat erillisiä moottoreita jokaista omalla ohjeellaan.

Tässä työssä taajuusmuuttajaa käsitellään neljään alijärjestelmään jaettuna järjestelmänä. Nämä ovat pääpiiri, ohjauselektroniikka, ohjelmisto ja lisäoptiot. Tämän lisäksi taajuusmuuttajan kehityksessä täytyy ottaa huomioon mekaaniset ominaisuudet, dokumentoinnin laatu, tuotanto, kuljetus, asennus ja lukuisia muita laitteen toteutusta tukevia toimintoja. Nämä on jätetty työn ulkopuolelle.



Kuva 2: Taajuusmuuttaja ja sen alijärjestelmät

Kuvassa 2 on esitetty taajuusmuuttajan lohkokaavio järjestelmäsuunnittelun kannalta. Taajuusmuuttajan pääpiiri kytketään sähköverkon ja sähkökoneen väliin, ja nämä toimivat järjestelmän jatkuvan toiminnan kuormana ja herätteenä tehon suunnasta riippuen. Pääpiirillä on rajapintoja ohjauselektroniikan kanssa, jota puolestaan käyttää taajuusmuuttajan laiteohjelmisto ja momenttisäätö. Ohjauselektroniikkaan on myös kytkettävissä turva- ja väyläoptioita, jotka ovat elektronisia piirikortteja ohjelmoituna rajattuun tehtävään tarkoitetuilla laiteohjelmistolla. Lisäksi taajuusmuuttajaan kytkeytyy mahdollisten optioiden tai suoraan ohjauselektroniikan kautta joukko sisään- ja ulostuloja, jotka toimivat myös järjestelmän herätteinä. Näin ollen järjestelmätestauksessa taajuusmuuttajan kiinnostavat rajapinnat ovat erityisesti Sähköverkko ja -kone, eri käyttöliittymät ja käyttötavat sekä taajuusmuuttajan toimintaympäristö. Näiden lisäksi taajuusmuuttajan sisäistä toimintaa joudutaan varmentamaan varsinkin järjestelmätestauksen alkuvaiheessa.

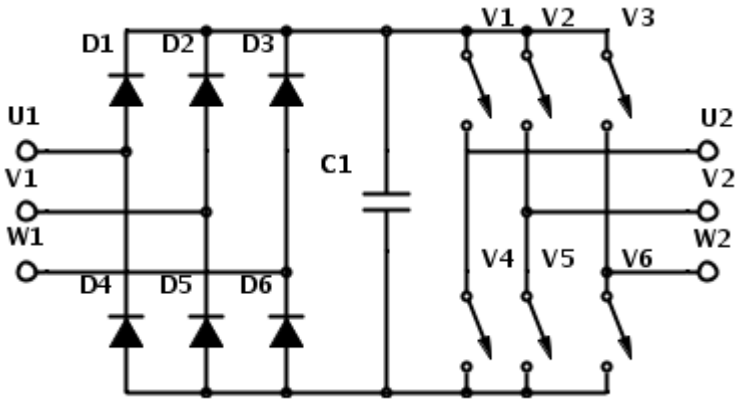
ABB Drivesin HPD-osastolla järjestelmän testausvastuu jakautuu alijärjestelmiä kehittävien yksiköiden ja erillisen testausyksikön kesken. Suunnittelusta vastaava yksikkö vastaa alijärjestelmien itsenäisen toiminnan testauksesta ja alijärjestelmän sisäisestä integroinnista täysin. Kun taajuusmuuttajan alijärjestelmiä integroidaan keskenään, jakautuu vastuu kunkin suunnitteluyksikön ja erillisen testausyksikön kesken vaihtelevasti niin, että suunnitteluyksiköt suorittavat molemmin puolin rajapintojen integrointitestejä ja testausyksikön tehtävänä on pääasiallisesti koko taajuusmuuttajan testaaminen, eli HPD:n tuotekehityksen järjestelmätestaus, sekä lopullinen

hyväksyntätestaus. Lähtökohtana on, että taajuusmuuttajan alijärjestelmät toimivat itsenäisesti ennen kuin kokonaisuutta testataan. Tässä kuitenkin on rajoituksia tapauksissa, jossa alijärjestelmä kohtaa todellisen käyttöympäristönsä vasta osana taajuusmuuttajaa, kun kuormituksena on todellinen sähkökäyttö ja -verkko. Näissä tapauksissa integrointitestaus ja järjestelmätestaus kulkevat lomittain tarpeen vaatiessa.

Seuraavaksi esitellään neljä taajuusmuuttajan alijärjestelmää, jotka liittyvät erityisen kiinteästi HPD:n tuotekehityksen sähköiseen testaukseen.

7.1 Pääpiiri

Taajuusmuuttajan pääpiiri on alijärjestelmä, jonka tehtävänä on laitteen tehonsiirto. HPD:n suunnittelemat taajuusmuuttajat ovat jännitevälipiirillisiä taajuusmuuttajia, joissa tulo- ja lähtösillan välisessä DC-välipiirissä käytetään energiavarastona kondensaattoreita.



Kuva 3: Jännitevälipiirillinen taajuusmuuttaja [5, osio 4 sivu 14 mukaillen]

Kuvassa 3 on esitetty jännitevälipiirillisen taajuusmuuttajan piirikaavio. Tulosilta D1 – D6 tasasuuntaa kolmivaiheisen verkkojännitteen U1, V1 ja W1, joka varastoidaan kondensaattoriin C1. Varastoitu DC-välipiirin sähköenergia siirretään lähtöpuolen puolijohdekytkimien V1 – V6 kautta muutetulla taajuudella ja amplitudilla lähtövaiheisiin U2, V2 ja W2. Lähtö on toteutettu IGBT-kytkimillä. Kytkimien rinnalla on lisäksi vastasuuntaan johtava paluudiodi, jonka kautta energia kulkee sähkökoneesta sähköverkkoon päin. Tulosilta on merkitty kuvaan 3 ohjaamattomana diodisiltana, mutta vaihtoehtoisesti voidaan käyttää tyristori- tai lähtöpuolen kaltaista IGBT-kytkentää, jos verkkopuolen toimintaa halutaan säätää. Kuvassa esitettyjen komponenttien lisäksi pääpiiriin kuuluu käytöstä riippuen muitakin komponentteja, kuten suodatukseen käytettyjä kondensaattoreita ja kuristimia.

Järjestelmäsuunnittelun näkökulmasta pääpiiri on virtatie, jonka ulostulon jännitteenmuotoa pitää pystyä ohjaamaan halutun taajuiseksi ja amplitudiseksi. Ulostulo riippuu käyttäjän haluamasta ajopisteestä, momenttisäädön laskemasta kytkennästä ja syöttöverkon jännitteenmuodosta. Pääpiirin suunnitteluun liittyvät esimerkiksi tehomitoitus, komponenttien oikea toiminta, nopeus ja luotettavuus eri käyttötilanteissa kuten eri kytkentä- ja lähtötaajuuksilla, hetkellisillä ja jatkuvilla tehoilla sekä vikatilanteissa.

Pääpiirin komponentteja testataan yksilöinä jolloin varmistutaan, että yksittäinen komponentti täyttää sille asetetut mitoitus- ja luotettavuusvaatimukset. Kun sopivat komponentit on valittu, testataan pääpiirin toimintaa kokonaisen järjestelmän osana kuormittamalla sitä taajuusmuuttajan nimellis- sekä muissa valikoiduissa pisteissä. Testien avulla pyritään määrittämään, toimiiko pääpiiri tarkoitetulla tavalla, pysyvätkö sen komponenttien toimintalämpötilat kestävällä tasolla ja jatkuuko pääpiirin toiminta stabiilina eri tilanteissa.

Erilaisia pääpiiriä kokonaisuutena testaavia testejä ovat esimerkiksi staattiset ja sykliset kuormitustestit, joissa mitataan muun muassa tehomitoituksen toimivuutta, ylivirta- ja ylijännitetestit suojauksen testaamiseksi sekä komponenttien hajoamistestit mahdollisen vikatilanteen turvallisuuden varmistamiseksi.

Pääpiiriä kokonaisuutena kuormittavia testejä tehdään paljon vasta järjestelmätestausvaiheessa, kun sen toimintaa ohjaa todellinen taajuusmuuttaja ja sillä on kuormana todellinen sähkökäyttö. Tätä ennen todellisen kuorman tarkka simuloiminen on haastavaa.

7.2 Ohjauselektroniikka

Ohjauselektroniikka on jaettu HPD:n taajuusmuuttajassa usealle ohjauskortille. Ohjauselektroniikan tehtäviin kuuluu toimia alustana taajuusmuuttajaohjelmistolle ja välittää muuttajan hilapulssit, joilla lähtökytkimiä ohjataan, IGBT-kytkimille kuorman ja käyttötilanteen mukaisesti. Lisäksi elektroniikka muuttaa ja välittää mittausdataa taajuusmuuttajan ohjaukselle sekä tuottaa ohjauskorttien itsensä sekä muun taajuusmuuttajan tarvitsemia jännitetasoja. Elektroniikkasuunnittelun vastuulla on HPD:n organisaatiossa myös suunnitella logiikkaohjelmisto, joka ohjaa piirikorttien toimintaa ja välittää dataa laiteohjelmiston ja elektroniikan välisellä rajapinnalla.

Ohjauskorttien testaaminen aloitetaan korttien suunnittelun ja prototyyppivalmistuksen jälkeen tutkimalla yksittäisen kortin toimintaa sen rajapintojen avulla. Testaus alkaa yksinkertaisista tapauksista kuten käyttöjännitteen syöttämisestä piiriin, jonka avulla nähdään käynnistyykö kortti halutusti. Tämän jälkeen kortin ja sen ohjauslogiikan toimintaa tutkitaan käyttämällä sen eri toimintoja mahdollisimman kattavasti. Tässä vaiheessa kortin testauksesta vastaa kortin suunnittelija.

Kun piirikorttien toimintaa on testattu itsenäisesti, yhdistellään kortteja pareittain suuremmiksi alijärjestelmiksi. Näin tutkitaan eri korttiyhdistelmien keskinäistä toimintaa ja laajennetaan alijärjestelmää vaiheittain, kunnes koko ohjauselektroniikan sisäinen toiminta on testattu.

Ohjauselektroniikan testaaminen ei ole täysin riippumatonta muista taajuusmuuttajan alijärjestelmistä. Ohjauskorttien mekaanisten ja sähköisten ominaisuuksien testaaminen onnistuu itsenäisesti erilaisilla testijärjestelyillä, mutta toiminnan oikeellisuuden testaamisessa tarvitaan hyvin nopeasti ohjelmistoa, joka käyttää elektroniikkaa ja pääpiiriä, jota elektroniikka ohjaa. Elektroniikkasuunnittelu saa ohjelmistosuunnittelulta taajuusmuuttajan laiteohjelmiston prototyyppejä testausta varten jo aikaisessa vaiheessa, mutta pääpiirin ja elektroniikan rajapinnan todellisen simuloinnin haastavuuden takia ohjauselektroniikka saadaan täysin todellisuutta vastaavaan kuormitukseen vasta kun koko taajuusmuuttajan prototyyppi on testauksessa.

7.3 Ohjelmisto

Taajuusmuuttajan ohjelmistokehitys on jaettu ABB:n organisaatiossa taajuusmuuttajan laiteohjelmiston (firmware) ja DTC-säädön (Direct Torque Control) suunnitteluun. Laiteohjelmisto käsittelee elektroniikan tuottamaa mittausdataa valvontaa ja ohjausta varten ja tarjoaa myös käyttöliittymän taajuusmuuttajalle. Säädön tehtävänä on puolestaan laskea IGBT-kytkimiä ohjaavat hilapulssit käyttötilanteen ja kuorman perusteella. Voidaan todeta, että laiteohjelmistossa toteutetaan taajuusmuuttajan korkeamman tason ohjaus kuten ajopisteen määrittely, toiminnan valvonta, jäähdytyksen ohjaus ja eri käynnistys- ja sammutustoimenpiteet ja säädössä matalan tason toiminta eli taajuusmuuttajan jatkuva kytkennän ja sähkönmuodon hallinta mahdollisimman tarkasti ja pienellä viiveellä. Molemmat ohjelmistosuunnittelun osat jakautuvat vielä suunnitteluorganisaatioiden sisäisesti ISU- (IGBT Supply Unit) ja INU-ohjelmiston (Inverter Unit) kehitykseen eli verkkopuolen muuttajan ja moottoripuolen muuttajan laiteohjelmiston ja säädön kehitykseen.

Ohjelmiston kehityksessä ABB:llä testaus tapahtuu kulloisenkin ohjelmiston rakenteen mukaisesti metodien ja funktioiden toiminnan testaamisella, jonka jälkeen aliohjelmia ja ohjelmia rakennetaan ja testataan yhä laajempina kokonaisuuksina. Sekä laiteohjelmiston, että säätöohjelmiston kehityksessä on koko ohjelmiston toiminnallisen testauksen käytössä ATF (Automated Tester Framework) -testiympäristö, jolla pystytään toteuttamaan automaattisesti ajettavia ja raportoituja testisekvenssejä. Myös HTR-testausympäristö, jonka ympärille tässä työssä kehitettävä testilaitteisto on rakennettu, perustuu ATF-ympäristöön.

ATF:llä ajetaan laiteohjelmiston kehityksessä monipuolisesti erilaisia ohjelmiston toimintapisteitä ja luetaan sen sisäisten muuttujien käyttäytymistä eri tilanteissa. Laiteohjelmistolle tehdään regressiotestauksessa tarvittavaksi koettu osa testirutiinista. Mahdolliset viat tallentuvat automaattisesti testiraporttiin, josta niitä voidaan tutkia tarkemmin ja jatkaa vianselvitystä manuaalisesti.

Laiteohjelmiston kehityskäytössä on useita pieniä moottorikäyttöjä, joilla testataan ohjelmiston toimintaa todellisessa laitteistossa. Testitapaukset liittyvät lähinnä ohjelmiston toimintaan ja varsinaisesti laitteistolähtöiset viat eivät ole testauksen painopiste.

Säädön kehityksessä ATF:llä ajetaan laiteohjelmiston testauksesta poikkeavia testitilanteita, joissa pääpaino on säädön muuttujien suorituskyvyn ja oikean toiminnan tarkkailulla. Valtaosa säädön regressiotestauksesta toteutetaan simuloidulla moottori- ja syöttöverkkomallilla, jolloin ympäristön epäideaalisuudet ja häiriöt eivät vaikuta testitulosten tulkitsemiseen ja kaikki poikkeukset ohjauksessa johtuvat säätöohjelmistosta. Simulointimallin lisäksi joitakin säätöohjelman testejä tehdään myös pienillä moottoritehoilla ja tarpeen vaatiessa vianselvitykseen käytetään myös suurempitehoisia ABB:n laboratorioissa olevia käyttöjä.

7.4 Optiokortit

Optiokortit ovat taajuusmuuttajaan liitettäviä laitteen ominaisuuksia laajentavia piirikortteja. Mahdollisia optioita ovat erilaiset käyttöliittymälaajennukset, joilla saadaan taajuusmuuttajan ohjauskortille lisää analogi- ja digitaaliliitäntöjä, kenttäväyläoptiot, joilla pystytään toteuttamaan sähkökäyttöjen ylempi ohjaus standardoitujen ohjausprotokollien kuten Modbus-, Profinet- tai Can-väylän välityksellä ja turvaoptio FSO (Functional Safety Option), jolla pystytään toteuttamaan sähkömoottorikäyttöjä koskevien standardien vaatimia turvatoimintoja, kuten pysäytyksiä ja pyörimisnopeuden rajoituksia tarvittaessa. [8]

Eri optioiden suunnittelu ja itsenäisen toiminnan testaaminen toteutetaan ABB:lla erillisissä yksiköissä. Option vastuuyksikkö suunnittelee optiokortin elektroniikan sekä laiteohjelmiston ja varmistaa, että sen rajapinnat sekä taajuusmuuttajan että ympäristön suhteen vastaavat haluttua. Erityisesti kenttäväyläoptioiden ja turvaoptioiden toiminta on vahvasti standardoitua, ja niiden ulkoisen rajapinnan toiminta suunnitellaan toteuttamaan rajapintaa koskevia standardeja.

Kenttäväyläprotokollia hallinnoivat järjestöt tarjoavat usein testaustyökaluja väylää käyttävien laitteiden standardinmukaisuuden testaamiseen, ja ABB:n kenttäväyläoptioita kehittävä yksikkö käyttääkin näitä työkaluja kenttäväylän puoleisen rajapinnan testaamiseen. Taajuusmuuttajaohjelman puoleinen geneerinen rajapinta täytyy testata omilla työkaluilla, mutta se on kaikille optioille yhteinen, joten samoja testitilanteita voi hyödyntää monilla optiokorteilla. Rajapintatestauksen lisäksi kenttäväyläoptioiden toimintaa testataan resurssien puitteissa yleisimpien taajuusmuuttajan ohjaukseen käytettyjen teollisuustietokoneiden ja ohjelmoitavien logiikoiden kanssa.

Turvaoptioiden kehityksessä optiokorttien toimintaa verrataan puolestaan teollisuuden turvastandardeihin ja asiakkaiden vaatimuksiin, jotka antavat tarkan aihion kehitystyölle. Turvastandardit antavat myös osan tarvittavista testitapauksista, jotka toteutetaan optiokorttien toiminnan testaamisessa. Muuhun suunnitteluun verrattuna turvaoptioiden suunnittelu on astetta muodollisempaa, ja turvaluokiteltuja komponentteja kehittäessä täytyykin kiinnittää erityistä huomiota raportoinnin sekä testaus- ja suunnitteluketjun aukottomuuteen.

8 Järjestelmätestauksen kehittäminen High Power Drivesilla

Tämän diplomityön aikana on haastateltu eri alijärjestelmiä kehittävien yksiköiden vastuuhenkilöitä ja muita avainhenkilöitä ja selvitetty, mitä eri alijärjestelmien ja koko järjestelmän näkökulmista voisi järjestelmätestausvaiheen toteutuksessa parantaa. Tämän perusteella pohditaan tässä luvussa vaihtoehtoja järjestelmätestauksen kehittämiseksi. Diplomityössä ehdotettuja kehityssuuntia voidaan käyttää tulevan testausstrategian suunnittelussa järjestelmätestauksen osalta. Tässä luvussa käsitellään osaltaan myös järjestelmätestauksen automatisointia, jota voidaan hyödyntää myös muissa testausvaiheissa.

Työn alkupuolella on selvitetty järjestelmätestauksen roolia testauksessa ja voitu huomata, että sen lähtökohtana oletetaan alijärjestelmien toimivan itsenäisinä kokonaisuuksina ja noudattavan rajapinnoillensa määriteltyjä ehtoja. Taajuusmuuttajan tapauksessa tämä tarkoittaisi, että pääpiiri, ohjauselektroniikka, ohjelmisto ja mahdolliset lisäoptiot toimivat itsenäisesti ja ovat integroitavissa ongelmitta keskenään.

Todellisessa tuotekehityksessä voi järjestelmätestauksen aikana paljastua vikoja, jotka johtuvat yksittäisen alijärjestelmän virheellisestä toiminnasta tai integraation ongelmista. Taajuusmuuttajan alijärjestelmistä osa on vaikeasti testattavissa täydellisesti ilman muiden alijärjestelmien toimintaa ja tämän takia myös osa alijärjestelmien testeistä saatetaan tehdä vasta järjestelmätestauksen ohessa. Tämä ei kuitenkaan saa olla järjestelmätestauksen pääpaino.

8.1 Automaattitestaus

Diplomityön varhaisessa vaiheessa päätettiin, että ensimmäinen askel järjestelmätestauksen kehittämiseksi on kehittää HPD:lle perusta automaattitestaukselle. Nykyisellään HPD:n testilaitteita täytyy testien aikana turvallisuussyistä valvoa useimmissa testitapauksissa paikan päällä, mikä kuluttaa paljon henkilöstöresursseja ja rajoittaa testausajan työpäivän tunteihin. Erityisesti hyväksyntätestausvaiheessa testejä toistetaan useissa pisteissä, jolloin periaatteessa samalla tavalla toistuvaan rutiinitestiin joudutaan käyttämään paljon työtunteja.

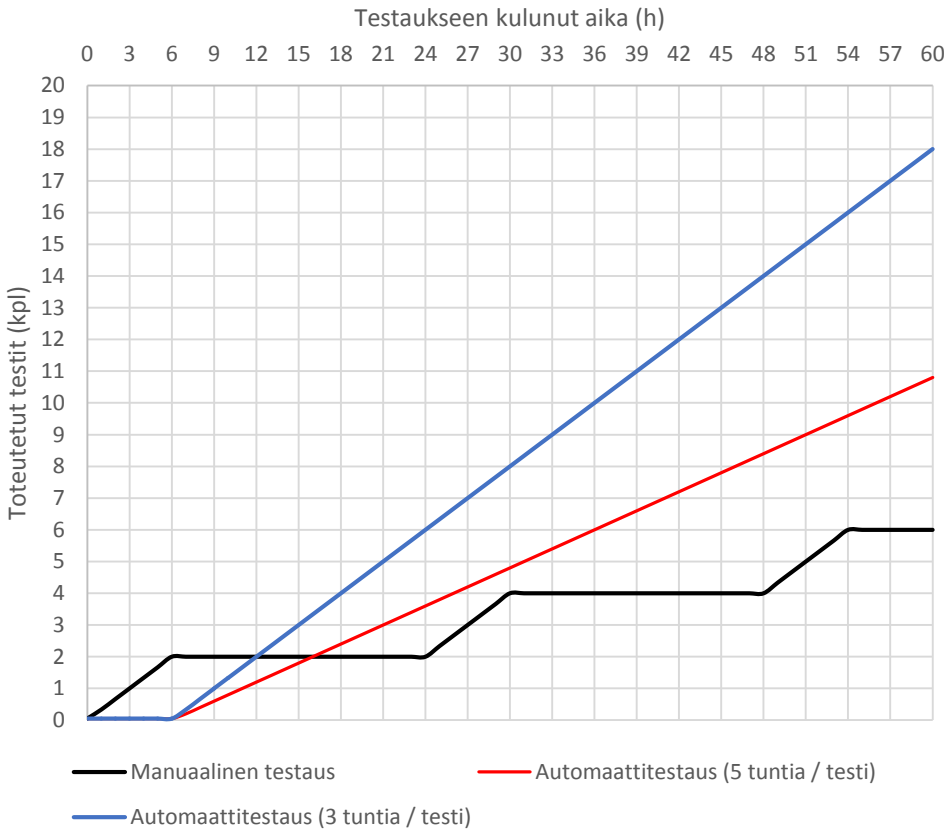
Automaattitestaus on hyvä tapa tehostaa erityisesti hyväksyntävaiheen testausta. Hyväksyntävaiheessa toistetaan testejä useissa eri toimintapisteissä [1]. Rutiininomaisessa testauksessa vakaasti toimivalle järjestelmälle automaattinen testausjärjestelmä tuottaa tuloksia manuaalista testausta paljon nopeammin [4].

Automaattitestausta suunniteltaessa täytyy varmistua, että automatisointi todella tehostaa testausta. Varsinkin käyttönsä alussa automaattinen testilaitteisto todennäköisesti aiheuttaa enemmän ongelmia kuin se ratkaisee [4]. Jos automaattisen testilaitteiston suunnittelee liian tiukasti määrättyyn testityyppiin, saattaa testilaitteisto vanhentua ja menettää merkityksensä suunnittelun edetessä muihin vaiheisiin tai ominaisuuden muuttuessa myöhemmässä kehitysvaiheessa. Automaattisesti toteutetun testausrutiinin muokkaaminen vaatii enemmän työtä kuin manuaalisen testin määrittelyn muuttaminen. [1]

Automaattista testilaitteistoa suunniteltaessa täytyy huomioida myös, että se vaatii testaajan rakentamaan testijärjestely, käyttöönottamaan sen ja testaamaan sen toimivuus, tulkitsemaan testin tuloksia ja tarvittaessa selvittämään mahdollisen vian syitä manuaalisesti. Lisäksi automaattinen testilaitteisto vaatii testaajilta erillistä koulutusta, ylläpitoa ja huoltoa. [4] Näin ollen automaattitestaus ei ole parhaimmillaankaan täysin ilmainen vaihtoehto manuaaliselle testaukselle.

Automatisointi vapauttaisi työaikaa HPD:n testauksessa erityisesti staattisten ja syklisten kuormitustestien tapauksessa, joissa taajuusmuuttajaa kuormitetaan määritellyllä teho- ja taajuuspisteellä, kuten esimerkiksi taajuusmuuttajan nimellisvirralla ja -lähtötaajuudella. Samalla mitataan taajuusmuuttajan kriittisten komponenttien lämpenemistä, jotta saadaan selville onko taajuusmuuttajan jäähdytys riittävä ja tehomitoitus onnistunut. Testeissä täytyy odottaa, että eri lämpömittaukset tasaantuvat kuormapisteessä, missä kestää useita tunteja. Erityisesti, jos testattavina komponentteina on suurikokoisia kuristimia, saattaa yhden työpäivän aikana pystyä testaamaan vain yhden kuormapisteen. Ottaen huomioon, että automaattista testausrutiinia pystyttäisiin parhaimmillaan jatkamaan ympäri vuorokauden, voitaisiin tämän kaltaisten testien tehokkuutta nostaa automatisoimalla moninkertaiseksi.

Automaattisella testausjärjestelyllä saavutettua työajan säästöä HPD:llä verrattuna manuaaliseen testaukseen voidaan hahmottaa arvioimalla molemmilla tavoilla toteutettujen testien määrää ajan funktiona. Kokemuksen perusteella voidaan arvioida kunkin testiajon kestoksi noin 3 tuntia. Todellisuudessa testien kesto saattaa vaihdella yhdestä viiteen tuntia välillä. Manuaalinen testaus vaatii täksi ajaksi yhden henkilön valvomaan testiä, minkä takia testejä voi suorittaa normaalisti vain kahdeksan tunnin työpäivän aikana. Arvioidaan lisäksi, että automaattisen testausjärjestelyn asennukseen ja konfigurointiin kuluu yhdeltä henkilöltä kuusi tuntia, minkä aikana testaus ei etene. Tämä aika sisältää testijärjestelyn komponenttien asennuksen, testisekvenssin konfiguroinnin ja sen toiminnan varmistamisen ajamalla se nopeutettuna läpi. Tätä vaihetta ei manuaalisessa testauksessa tarvitse suorittaa. Testattavan taajuusmuuttajan ja tarvittavien mittausten asennusta ei oteta arviossa huomioon, sillä ne ovat molemmille testaustavoille yhteisiä toimenpiteitä, eivätkä vaikuta testaustapojen keskinäiseen nopeuteen. Yllä mainittujen arvioiden perusteella voidaan piirtää kuvaaja eri testaustapojen toteutuksesta.



Kuva 4: Kuormitustestien toteutukseen kuluva aika manuaalisesti ja automaattisesti testaamalla

Kuvassa 4 on esitettynä kuormitustesteihin kuluva aika eri testaustavoilla yllä kuvailtujen oletusten perusteella. Manuaalisesti testattuna kyetään vuorokaudessa testaamaan kaksi määriteltyä pistettä, minkä jälkeen testaus on pysähdyksissä 18 tuntia. 54 tunnissa olisi näin testattu kuusi pistettä. Automaattitestauksella ensimmäiset kuusi tuntia kuluvat testijärjestelyn ylimääräisessä asettelussa, mutta sen jälkeen kaikki määritellyt testit suoritetaan parhaassa tapauksessa ilman katkoksia. Taulukkoon on merkitty automaattitestauksen kohdalla kaksi arviota. Sinisellä janalla merkityssä tilanteessa automaattitesti toteuttaa testipisteet samassa kolmen tunnin ajassa kuin manuaalinenkin, mikä tarkoittaa että testinvalvonta kykenee havaitsemaan lämpötilojen tasaantumisen tarvittavalla tarkkuudella heti, kun se tapahtuu. Ihmistestaaja kykenee arvioimaan tämän, vaikka mittauksissa olisikin ongelmia. Punaisella janalla on arvioitu pahinta mahdollista tilannetta, jossa mittausten kohina tai muu vika estää automaattista testinvalvontaa toteamasta testipisteen lämpötilojen tasaantumisen. Tällöin jokaiseen testipisteeseen kuluva aika olisi testivaiheen pituudelle asetettu takaraja, joka täytyy asetella suurimman mahdollisen keston mukaan. Yllä olevien arvioiden perusteella tämä takaraja olisi viisi tuntia.

Arvioiden perusteella automaattinen testaus toteuttaisi testit manuaalista nopeammin, jos määritellään yli kaksi testipistettä. Tämä pätee sekä kolmen, että viiden tunnin testipisteen kestolla. Molemmissa tilanteissa kolmas testipiste toteutuisi ensimmäisen vuorokauden aikana, kun manuaalisessa testauksessa on valvonnan tarpeen takia tauko. Automaattisella testauksella saavutettu etu myös kasvaa sen mukaan, mitä useampia testipisteitä määritellään. Ottaen huomioon yllä olevan lisäksi sen, että automaattinen

testaus ei tarvitse ihmistä valvomaan testiä testin ajaksi, saavutetaan automaattitestauksessa henkilötyötunteina vielä suuremmat edut. Tässä arviossa ei toki ole otettu huomioon vikaantumis- ja ongelmatilanteita, jolloin kumman tahansa testaustavan testaukseen kuluva aika kasvaa odottamattomasti. Tällöin manuaalisen testauksen etu on kuitenkin se, että vika havaitaan välittömästi ja siihen kyetään puuttumaan niin, että vikaantumishetken olosuhteet ovat suoraan testaajan tiedossa. Automaattitestauksessa vian tapahtumahetkestä kerättävä tieto on testausjärjestelyn diagnostiikan varassa, minkä takia sen tasoon pitääkin kiinnittää erityisen paljon huomiota.

Tilanteissa, joissa testauksen automatisointi kannattaa, pitää se tehdä testauksen laadullisia ja turvallisuuteen liittyviä periaatteita noudattaen. Automaattitestilaitteistoa suunnitellessa täytyy ottaa huomioon testijärjestelyn luotettavuus, turvallisuus ja joustavuus. Testijärjestelmän pitää luotettavasti ajaa haluttua testirutiinia ja tallentaa itsenäisesti ja varmasti kaikki ajon aikana kerättävä tieto. Jos testi katkeaa odottamattomasti, pitää testilaitteiston tallentaa katkeamiseen mennessä kerätty data, jotta voidaan selvittää testin katkeamisen syyt. Turvallisuuden kannalta testilaitteiston pitää pysäyttää testattava laite ja estää sen uudelleenkäynnistyminen, jos laite käyttäytyy vaarallisesti. Suunniteltava testijärjestely saattaa aiheuttaa vaaraa tai tuhoutua esimerkiksi ylikuumenemalla, syttymällä palamaan tai ohjaamalla taajuusmuuttajan ja sitä kautta moottorin vaarallisille ohjausalueille. Myös erilaiset ilmoitukset tai hälytykset vian tapahtuessa, testin eristäminen ja muu ympäristön turvaaminen liittyvät testilaitteiston turvallisuuteen. Testilaitteiston joustavuus tarkoittaa, että se on käytettävissä mahdollisissa eri tilanteissa ja laajennettavissa tarvittaessa uusiin käyttöihin. Tämän varmistamiseksi testilaitteisto suunnitellaan käyttöönsä mahdollisimman yksinkertaiseksi ja yleisluontoiseksi, jotta se on muokattavissa myös alkuperäistä suunnitelmaa laajempaan tai siitä poikkeavaan käyttöön niin haluttaessa.

8.2 Testauksen kattavuuden kehittäminen

Jotta automaattitestaus parantaisi testauksen laatua, täytyy vapautuva testausaika käyttää testauksen laadun parantamiseen. Tätä varten täytyy pohtia uusia testausmenetelmiä, jotka antavat uutta tarpeellista tietoa taajuusmuuttajan toiminnasta. Tällaisia testitilanteita on selvitetty asiantuntijahaastattelujen perusteella.

Automaattitestilaitteiston toimiessa riittävän hyvin on lisääntyvää testausaikaa helppo käyttää aiempaa kattavampaan testipisteiden läpikäymiseen kuormitustestissä. Jo pelkästään kehitettävien taajuusmuuttajien ajoajan lisääntyminen tuo lisää rasitusta laitteille ja tuo ilmi mahdollisia ongelmia. Ajopisteiden monipuolistaminen mahdollistaa toiminnan lineaarisuuden varmistumisen aiempaa tarkemmin. Erityisesti korkeat IGBT-kytkinten kytkentätaajuudet, syöttö- ja lähtöverkon eri taajuudet sekä erilaiset syklit ovat mielenkiintoisia ja mahdollisesti haastavia toimintapisteitä taajuusmuuttajalle. Näistä testipisteistä korkeiden verkon taajuuksien testaaminen vaatii sähkökoneita, jotka toimivat halutulla taajuudella. Todella korkeilla pyörimisnopeuksilla toimivia koneita ei toistaiseksi ole testauskäytössä, joten tämä vaatisi investointia uuteen testilaitteistoon.

Haastavat kuormituspisteet kannattaa selvittää ja suunnitella niiden perusteella testausrutiini automaattitestilaitteelle. Jos haastavimpien pisteiden ja uudempien prototyyppien tapauksessa testiin liittyy erityisiä riskejä, kannattaa se toteuttaa valvottuna. Automaattitestilaitteisto saattaa nopeuttaa näissäkin tapauksissa mittausdatan

keräämistä ainakin, jos mittausdatan muoto saadaan tallennettua sopivaan muotoon. Pääasiallisesti valvomattomia testejä kannattaa suorittaa testipisteissä, joissa ei odoteta tapahtuvan fataaleja ongelmia.

Taajuusmuuttajan momenttisäädön testauksen kannalta kuormitusajoihin olisi hyvä liittää taajuusmuuttajan dynaamisen toiminnan testausta. Säädön testejä tehdään nykyään paljon simuloimalla sekä pienemmillä moottoritehoilla, joten osan moottorisäädön testeistä toistaminen suurempitehoisilla käytöillä varmistaisi säädön laatua. Varsinkin mahdollisissa vianselvitystilanteissa suurempitehoisen käytön saatavuus on aikaisemmin voinut olla haastavaa. Jos käytettävissä olevaa testausaikaa saadaan lisättyä tarpeeksi, voi testausaikaa olla hyödyllistä käyttää säädön suorituskyvyn testaamiseen.

Taajuusmuuttajan ohjaaman moottorin kuorman muutokset, momentin tarkkuuden testaaminen eri pisteissä, kiihdytys- ja jarrutusrampit sekä taajuusmuuttajan toiminta nollanopeuden ympäristössä ovat momenttisäädölle oleellisia pisteitä. Nollanopeuden lähistö ja ramppiajat ovat myös FSO-turvaoption valvontaan käyttämiä rajaehtoja. Näitä testejä voisi liittää taajuusmuuttajan mekaanisia ominaisuuksia ja tehomitoitusta tutkivien testien joukkoon ja testattavaksi jopa samalla testillä. Pääpiirin tehomitoitusta tutkiessa voitaisiin mitata myös momentin tarkkuutta ja lisätä syklisiä kuormitusajoja, joissa taajuusmuuttajalla ajetaan kiihdytys- ja jarrutusramppeja. Momentin tarkkuuteen liittyvät testit tosin vaativat testauslinjaston moottorin akseliin momenttianturia, jotta taajuusmuuttajan mittaamalle sähköiselle momentille saadaan vertailukohta. Momenttiantureita ei ole asennettu useimpiin suuritehoisiin moottorilinjastoihin, joten niihin pitäisi näitä testejä varten investoida.

Taajuusmuuttajan dynaamisen toiminnan lisäksi moottorisäädölle mielenkiintoisia ovat käytöt, joissa moottorikaapelit ovat pitkät suhteessa lähtötaajuuteen. Pitkät moottorikaapelit saattavat aiheuttaa värähtelyä ja heijastuksia, jotka vaikuttavat säädön tarkkuuteen. Tällaisten testien lisääminen voi myös olla hyödyllistä. Pitkiä moottorikaapeleita on asennettu kiinteästi muutamaan testipisteeseen, mutta niiden käyttöä voisi lisätä.

Usean alijärjestelmän kannalta kiinnostava testi on rinnankäyvän taajuusmuuttajan testaaminen niin, että yksi tai useampi rinnankäyvistä moduuleista lopettaa toimintansa. Taajuusmuuttajaohjelma tarjoaa mahdollisuuden tällaiseen käyttöön, ja käyttötavan kattava testaus olisi hyödyllistä. Taajuusmuuttajan säätö ja pääpiiri puolestaan joutuvat toimimaan tällöin normaalikäytöstä poikkeavassa tilanteessa. Mahdollisia ongelmakohtia ovat esimerkiksi taajuusmuuttajan virtarajat, joiden pitäisi yhden moduulin puuttuessa pienentyä normaalitilanteeseen verrattuna sekä yhden moduulin mittausdatan puuttuminen säädöltä, mistä huolimatta järjestelmän pitäisi jatkaa toimintaa ongelmitta.

Laajat ja monimutkaiset käytöt saattavat olla taajuusmuuttajalle haastavia. Eri toimintojen vasteajat ja keskinäiset viiveet saattavat aiheuttaa ongelmia erityisesti, kun sähkökäyttö sisältää pitkiä signaaliteitä ja useita taajuusmuuttajia. Yksi tällainen tilanne on Master – Follower -käyttö. Master – Follower -käytössä yksi laite ohjaa järjestelmän toimintaa ja välittää tilansa perusteella ohjeen seuraajilleen [9]. Tämä käyttötapa on ABB:lla mahdollinen sekä taajuusmuuttajan ohjauksessa [9] että FSO-turvaoptioiden keskinäisessä linkityksessä [10]. Erityisesti ohjauskatkot, häiriöt ja ohjaussignaalin katoaminen matkalla saattavat aiheuttaa ongelmia monimutkaisissa järjestelmissä.

Testikäytöt ovat usein maksimissaan muutamasta käytöstä koostuvia, mutta laajimman tehtävissä olevan käytön toiminnallinen ja suorituskykytestaaminen kommunikaatioväylien ja prosessorien maksimikuormilla olisi hyödyllistä. Erityisesti huomioitava testaustapa on taajuusmuuttajan käynnistämisen ja sammuttamisen toistaminen, koska erityisesti monia taajuusmuuttajia sisältävissä käytöissä käynnistyksenaikaisten alustusten ajoituksessa saattaa olla ongelmia. Tämä aiheuttaa turhaa uudelleenkäynnistystä ja ylimääräisiä virheitä yhden alijärjestelmän ollessa eri vaiheessa käynnistyssekvenssiä kuin toinen alijärjestelmä olettaa.

Laajimmat testauskäytössä olevat käytöt lienevät jatkossakin pilottiasiakkailla, jotka ottavat uuden sukupolven taajuusmuuttajia todelliseen käyttöön tuotekehitysvaiheessa. Pilottikäyttöjenkin on tarkoitus toimia vähintään tyydyttävästi, jotta asiakas ei turhaudu uuteen laitteeseen. Tämän takia mahdollisimman monimutkaisten taajuusmuuttajakokonaisuuksien testaaminen ABB:n sisäisesti on pilottitestauksen käytöstä huolimatta kannattavaa.

Kenttäväyläohjaus on nykyään usein käytetty taajuusmuuttajan ohjaustapa. Tämän takia voidaan perustella, että taajuusmuuttajan kommunikointi kenttäväylän kautta taajuusmuuttajaa ohjaavan ohjelmoitavan logiikan kanssa on osa järjestelmän toimintaa. Kenttäväyläoptioita suunnitteleva yksikkö on vastuussa kenttäväylän toiminnan testaamisesta ja standardinmukaisuudesta, mutta järjestelmätestauksen kannalta voi olla tarpeellista testata taajuusmuuttajan ohjattavuus ja sen viiveet erityisesti suurella prosessorikuormalla ja muissa ohjaukselle haastavissa ympäristöissä.

Testausajan rajallisuuden takia on testattavista taajuusmuuttajista aiemmin karsittu paljon ei-kriittisiä ominaisuuksia testille oleellisen toiminnan varmistamiseksi. Jos yksinkertaisimmat testit saadaan automatisoitua, voi koko järjestelmän kannalta olla hyödyllistä käyttää testaajien aikaa lisäominaisuuksien integroimiseen nykyistä suurempaan osaan rakennetuista testijärjestelyistä ja lisäominaisuuksien perustoiminnan testaaminen useissa eri käytöissä. Esimerkiksi erilaiset turvatoiminnot ja laiteohjelmiston asetukset ovat sellaisia, joiden laajempi käyttö eri tilanteissa voi tuoda ilmi satunnaisesti ilmeneviä vikoja.

Alijärjestelmät on testattu itsenäisesti toimivina kokonaisuuksina, mutta niiden välisissä väylissä tapahtuvien ongelmien käsittelyä saattaa olla hyvä lisätä. Eri valvonta- ja ohjaussignaaliteiden katkokset ja häiriöt saattavat paljastaa vielä järjestelmävaiheessa tilanteita, joita alijärjestelmät eivät ole aikaisemmassa testauksessa kokeneet. Uudessa laitesukupolvessa on esimerkiksi vaihdettu joitakin aiemmin valokuidulla toteutettuja signaaliteitä Ethernet-kaapeleilla, minkä tuloksena taajuusmuuttajaan on syntynyt täysin uusi kytkeytymisväylä häiriöille.

Eri mittapiirit saattavat olla herkkiä maatason häiriöille. Tämän testaaminen voi olla eduksi. Elektroniikkapiirien kannalta maasulkutilanteet ja muuten huonot maadoitukset ovat raskas toimintaympäristö, joten testaamisen lisääminen häiriöisellä maapotentiaalilla saattaa paljastaa ongelmia immuniteetissa erityisesti pitkien testiajojen aikana. Myös sähköverkon yliaallot ja radiotaajuiset piikit voivat tuoda ilmi ongelmia järjestelmän toiminnassa. Elektroniikan lisäksi häiriöinen ympäristö testaa laite- ja säätöohjelmistojen häiriönkäsittelyä ja suodatuksen toimivuutta. Kokonainen taajuusmuuttajakäyttö on ensimmäinen tilanne, jossa kaikki alijärjestelmät kohtaavat todellisen ympäristön, joten ympäristön suunnitteleminen pahimman mahdollisen käytön

mukaan myös kosteuden, lämmön ja muiden ympäristöllisten tekijöiden suhteen testaisi taajuusmuuttajan ääritilanteiden kestoa. Piirikortit ovat esimerkiksi herkkiä korkeille ympäristön lämpötiloille, joiden käyttö testeissä saattaa tuoda ilmi vikoja.

Regressio- ja yhteensopivuustestauksen systemaattinen toteutus on yksi oleellinen kehityssuunta. Taajuusmuuttaja koostuu useista eri alijärjestelmistä, joiden versiot muuttuvat tuotekehityksen aikana useaan otteeseen. Uusien versioiden integrointi vanhoihin testilaitteisiin voi aiheuttaa odottamattomia yhteensopivuusongelmia. Jos järjestelmälle kehitettäisiin regressiotestaussuunnitelma eri alijärjestelmien muutosten varalle, voitaisiin tällainen toiminnallinen testirutiini toteuttaa aina versiopäivityksen yhteydessä alijärjestelmän itsenäisen testin lisäksi koko järjestelmällä. Tällainen toimintatapa saattaisi sopivasti toteutettuna paljastaa vikoja, jotka eivät alijärjestelmien itsenäisissä testeissä paljastu. Versiopäivityksen onnistumista voidaan varmistaa myös parantamalla kommunikaatiota. Muutoksen tekijän olisi hyvä ilmoittaa alijärjestelmän sidosryhmille kuten testausyksikölle ja alijärjestelmän kanssa rajapinnan jakaville suunnitteluyksiköille muutoksen mahdollisista vaikutuksista rajapintoihin sekä odotetusta testaustarpeesta. Jo pelkästään testausvastuun selkeyttäminen alijärjestelmän muutoksen yhteydessä parantaisi testauksen laatua.

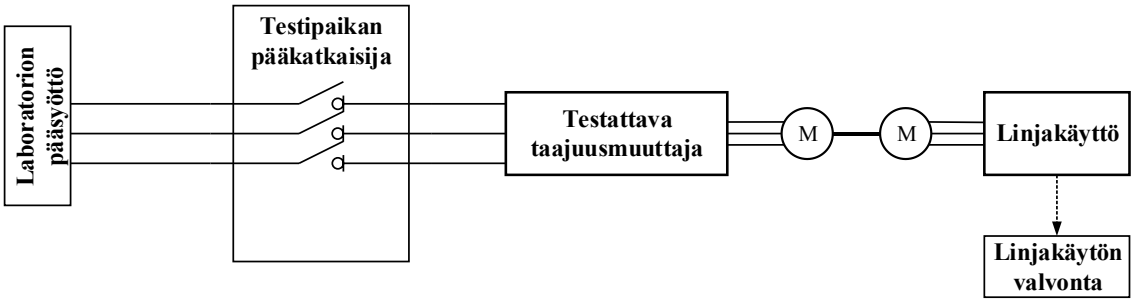
Taajuusmuuttajan suojarajat riippuvat muuttajan teholuokituksesta. Eri teholuokituksia on kymmeniä, ja jokaisella näistä on oma luokitustiedostonsa, jossa rajat ja muut muuttajan tehoon liittyvät parametrit määritellään. Näissä määrittelyissä saattaa piillä harvoin käytössä olevia rajoja, joihin täytyy kiinnittää erityistä huomiota. Luokitustiedostojen yhteensopivuuden kattavaan testaamiseen voi olla hyvä kiinnittää huomiota yhteensopivuustestausta suunniteltaessa.

Luotettavuustestausta voi lisätä aina resurssien mukaan. Erilaiset piirikortteja ja pääpiirin komponentteja vanhentavat kiihdytetyt ja kiihdyttämättömät elinikätestit tuovat ilmi taajuusmuuttajan heikoimpia komponentteja ja ennakoivat elinikää niin, että vanheneminen ei asiakaskäytössä tule yllätyksenä ja aiheuta tyytymättömyyttä. Todella kriittiset luotettavuusongelmat paljastuvat kiihdytetyissä testeissä nopeasti, ja mitä enemmän ja suuremmilla laitemäärillä testausta toteuttaa, sitä luotettavammaksi taajuusmuuttaja voidaan todeta. Tuotekehitysvaiheessa, kun taajuusmuuttajakokoonpanojen määrä on suhteellisen pieni, on kattavan luotettavuustestauksen toteuttaminen haastavaa. Pienemmätkin elinikätestaukselle altistetut testierät kuitenkin auttavat varmentamaan luotettavuutta. Taajuusmuuttajan komponentit vanhenevat suurelta osin ympäristön ja sähköisten komponenttien häviötehon aiheuttaman lämpötilan seurauksena, joten testipisteen teho ja ympäristön lämpötila ovat luontevia kiihdytystekijöitä luotettavuustesteissä.

Yksi laitteen elinikään vaikuttava tekijä on sen kokemat vikatilanteet. Jo nykyään testataan turvallisuussyistä taajuusmuuttajien ylivirta-, ylijännite- ja ylilämpövikojen hallintaa sekä erilaisia suoria ja valokaarellisia oikosulkuja. Näiden suojauksien toiminta on testattu, mutta voi olla hyödyllistä resurssien salliessa lisätä pääpiiriä kuormittavien vikatilanteiden toiston määriä, jotta saadaan selville kuinka paljon viat taajuusmuuttajaa vanhentavat. Esimerkiksi jokainen oikosulku aiheuttaa hetkellisen ylivirtapiikin, joka saattaa vaikuttaa laitteen komponenttien elinikään.

9 Automaattitestauksen järjestäminen

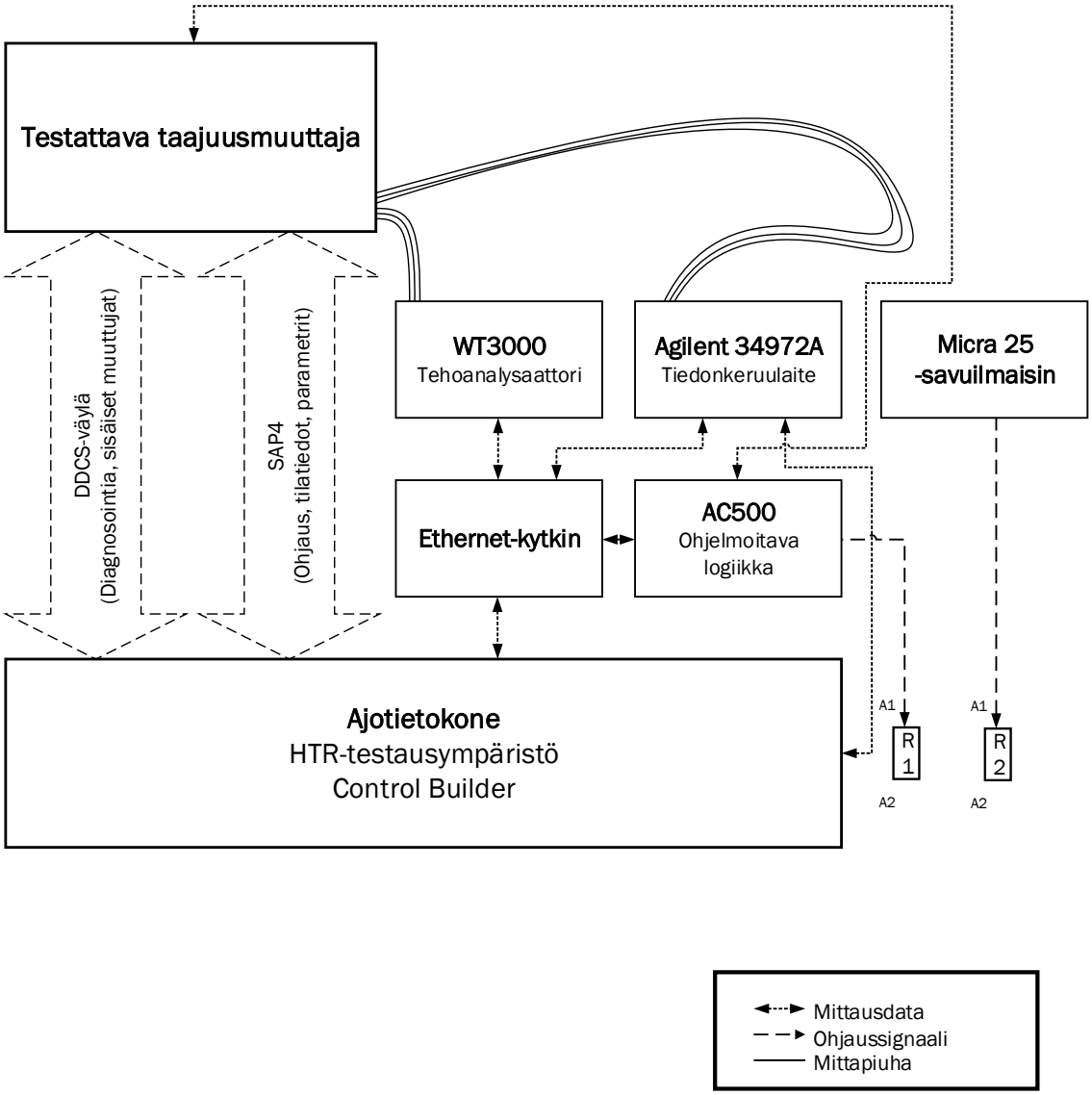
Tässä luvussa esitellään tämän diplomityön aikana kehitetty automaattitestilaitteisto. Testilaitteisto koostuu tietokoneella toimivasta HTR-testausympäristöstä, joka ohjaa ja valvoo taajuusmuuttajan toimintaa testin aikana, sekä mittalaitedataa keräävästä AC500-logiikkaohjaimesta. HTR on ollut ABB:n LAC-osaston käytössä jo useampia vuosia, mutta HPD-osaston testauksessa sitä ei ole vielä käytetty. Tästä johtuen HTR:ää ei tässä työssä muokattu lainkaan, vaan se on pelkästään otettu käyttöön. AC500-logiikkaohjelman pohja on myös saatu LAC:lla toteutetusta testausjärjestelystä, mutta sille tehtiin tässä työssä useita muutoksia jotta se soveltuisi laajempaan ja mukautuvampaan käyttöön.



Kuva 5: ABB:llä yleisesti käytetty taajuusmuuttajan testijärjestely

Kuvassa 5 on esitetty ABB:n Helsingin tehtaassa yleisesti käytetty testijärjestely. Järjestelyssä testattava taajuusmuuttaja ohjaa moottoria, jolla on yhteinen akseli testilaitetta kuormittavan sähkökäytön kanssa. Kuormituksen toteuttava linjakäyttö koostuu sähkömoottorista ja tuotannossa olevasta aikaisemman sukupolven taajuusmuuttajasta. Kuormittava käyttö jarruttaa testattavan käytön tuottamaa tehoa takaisin sähköverkkoon. Tällä testijärjestelyllä pystyy ajamaan eri nopeus- ja virtapisteitä ja muokkaaman taajuusmuuttajan parametreja testipisteen tarpeen mukaan sekä testattavasta taajuusmuuttajasta että linjakäytöstä. Muokattavia ominaisuuksia ovat pisteessä käytettävien pyörimisnopeuden ja tehon lisäksi esimerkiksi IGBT-kytkinten kytkentätaajuus, eri turvarajat ja -toiminnot.

HPD:n testeissä on taajuusmuuttajaa tähän asti ohjattu lähinnä manuaalisesti ohjauspaneelilla tai tietokoneohjelmalla. Rajoittavia tekijöitä automaattitestauksen käyttöönotossa ovat olleet testien ohjauksen toteutus sekä palo- ja henkilöturvallisuus. Tässä työssä pyritäänkin toteuttamaan automaattitestilaitteisto, joka ohjaa laitteen toimintaa, mittaa testin lämpötiloja ja sähköisiä suureita itsenäisesti ja luotettavasti sekä pysäyttää testin vikatilanteessa luotettavasti.



Kuva 6: Suunniteltu automaattitestilaitteisto

Kuvassa 6 on esitetty suunniteltu automaattinen testauslaitteisto kokonaisuudessaan. Järjestelmän ytimenä toimii Windows 7 -käyttöjärjestelmällä varustettu ajotietokone, johon on asennettu HTR-testausympäristö ja AC500-logiikan ohjelmointiin käytettävä Control Builder -ohjelmointiympäristö. Ajotietokone on yhdistetty taajuusmuuttajan ohjauskorttiin valokuitulinkin (DDCS) ja ohjauspaneelin mini-USB-liitännän (SAP4) välityksellä. Näiden väylien kautta tapahtuu kaikki HTR:n kommunikointi taajuusmuuttajan kanssa. Myös yksi Agilent 34972A -tiedonkeruulaite on kytkettävissä ajotietokoneeseen USB-väylän kautta, jolloin HTR pystyy käyttämään sen lämpömittauksia testin aikana. AC500 on yhdistetty ajotietokoneeseen ja käytettyihin mittalaitteisiin, eli Yokogawan WT3000 tehoanalysaattoriin ja 1 – 3 34972A-tiedonkeruulaitteeseen Ethernet-kytkimen kautta RJ45-kaapeleilla. AC500 kerää mittausdatan ja tallentaa sen tiedostoksi SD-muistikortille. Lisäksi AC500 kommunikoi Modbus-protokollalla taajuusmuuttajan kanssa Ehternet-verkon välityksellä.

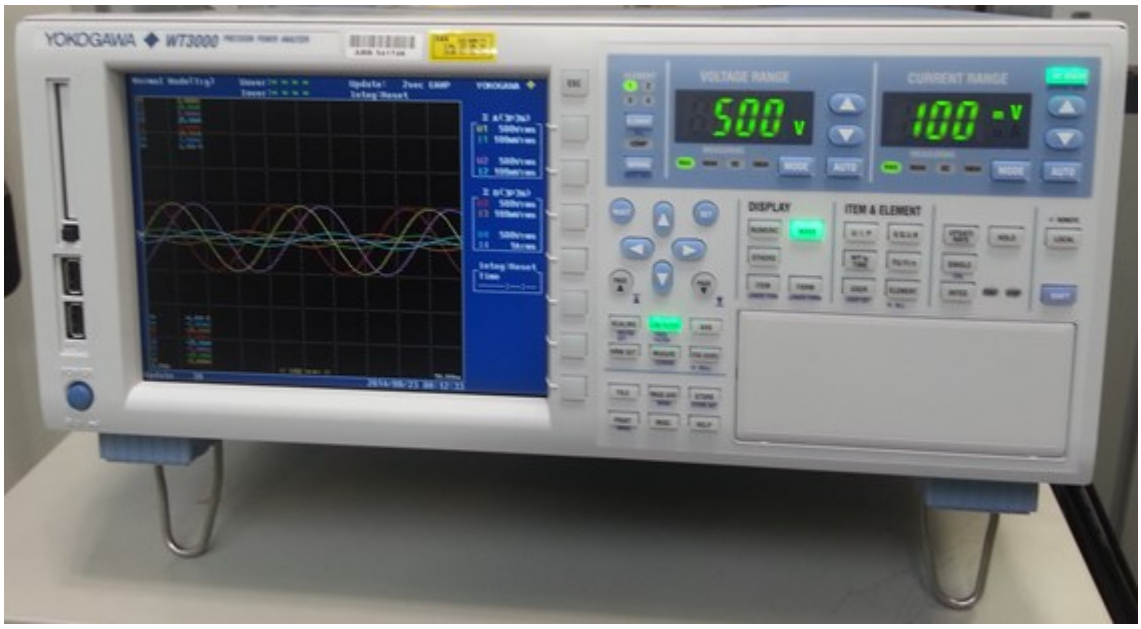
Testijärjestelyn paloturvallisuuden parantamiseksi laitteistoon lisätään Micra 25 -savuilmaisin, jossa on ilmaisimen lauetessa toimiva relelähtö [11].

Kuva 7: Agilent 34972A -tiedonkeruulaite

Agilent 34972A (kuva 7) on tiedonkeruulaite, jolla pystytään keräämään mittausdataa erilaisista sähköisistä suureista. Agilent 34972A:n vahvuus on mittauksen herkkyys sekä mahdollisuus mitata kymmeniä mittauksia samanaikaisesti. Näytteenottotaajuus on puolestaan laitteen heikkoja puolia. ABB:lla 34972A on käytössä pääasiassa lämpömittauksissa termoparien ja vastuslämpömittareiden kanssa. [12]

Kuva 8: Agilent 34901A -multiplekserimoduuli

Tiedonkeruulaite mahdollistaa kolmen 34901A-multiplekserimoduulin (kuva 8) käytön. Jokaisessa moduulissa on 20 ruuvikiinnityksellä liitettävää mittauskanavaa jännitteelle tai vastukselle ja referenssilämpötilamittaus lämpömittauksia varten. Lämpöantureita saa tiedonkeruulaitteeseen liitettyä yhteensä 60 kappaletta. [12]



Kuva 9: WT3000-tehoanalysaattori

Yokogawa WT3000 (kuva 9) on tehoanalysaattori. WT3000:ssa on neljä eristettyä sisääntuloelementtiä, joihin jokaiseen pystyy liittämään yhden jännitemittauksen ja yhden ulkopuolisen virtamittauksen, jonka antama mittaustulos WT3000:lle on jännitemuotoinen. Neljää elementtiä voi käyttää erillisiin mittauksiin tai niiden tuloksia voi käyttää yhdessä useampivaiheisiin mittauksiin, kuten kolmivaiheverkon tehomittaukseen kahta jännite- ja kahta virtamittausta käyttävällä niin kutsutulla Aaron-kytkennällä. [13] WT3000:lla on mahdollista suorittaa mittauksille teho-, vaihe, yliaalto- ja paljon muuta laskentaa käyttäjän niin halutessa. Yhteen WT3000-tehoanalysaattoriin pystyy kytkemään joko taajuusmuuttajan tulo- ja lähtöpuolet Aaron-kytkennällä tai toisen tulo- ja lähtöpuolesta sekä DC-välipiirin virta- ja jännitemittauksen niin halutessa. Molemmat mittaustavat ovat käytössä HPD:llä.

9.1 Heat Test Runner

Heat Test Runner (HTR) on ABB:lla kehitetty testausympäristö. Se on suunniteltu ajamaan erilaisia taajuusmuuttajan kuormitustestejä ennalta määritellyn testisekvenssin mukaisesti. Testausympäristö pystyy muuttamaan ja lukemaan testattavan taajuusmuuttajan sekä taajuusmuuttajaa kuormittavan linjakäytön laiteohjelmiston parametreja sekä ohjelman sisäisiä muuttujia [14], joten teoriassa HTR:n kykenee kaikkiin testipisteisiin, joita manuaalisestikin pystyisi ajamaan. Täysin uuden testityypin kehittäminen vaatii kuitenkin jonkin verran ohjelmointia C#-ohjelmointikielellä suoraan HTR:n ohjelmakoodiin. HTR:ään on tämän takia käytön helpottamiseksi implementoitu muutama yleisin testityyppi, joita voi yhdistellä haluamallaan tavalla testisekvenssiksi. Näistä käytetään tässä työssä kehitettävässä testijärjestelyssä lähinnä staattista kuormitusajoa.

HTR:ään on tällä hetkellä mahdollista kytkeä yksi 34972A-tiedonkeruulaite, jota käytetään testeissä lämpötilojen mittaamiseen. HTR asettelee Excel-tiedoston perusteella 34972A:n mittauskanavat ja mittaa niiden lämpötiloja asetuksen mukaan suodattamattomana tai suodatettuna. Jokaiselle kanavalle on mahdollista määrittää varoitus- ja hälytysraja, joiden perusteella HTR reagoi liian korkeisiin lämpötiloihin ja

hälytysrajan ylittyessä pysäyttää testin. Lisäksi kanavien lämpötilojen tasaantumisen pystyy määrittämään testin loppumisehdoksi. [14]

HTR:n kommunikointi testattavan ja kuormittavan taajuusmuuttajan sekä tiedonkeruulaitteen kanssa määritellään Excel-tiedostoon HTR-ohjelman kansiossa. Tiedostoon kirjoitetaan eri kommunikointitavat, joita eri laitteet käyttävät sekä parametrit, muuttujat ja lämpömittaukset, jotka halutaan tallentaa.

# Item	DataType	ParamName	MinLimit	MaxLimit	DisplayFormat	Unit	ByteSize	Actions
102	33095681	Motor speed rpm	-1600	1600	0	rpm	4	N:SPEED
110	33095681	Motor torque	-100	100	0	%	4	M:N
107	33095681	Motor current		1500	0	A	4	M:N:PID
111	33095681	DC voltage	500	1100	0	V	4	N

Kuva 10: HTR:n valvomien parametrien määrittely

# DLChannel	ProbeType	TempFilter	TempLimits[C]	gHead	Actions
101	2wRTD85	MAV:3	100:150	1	T:EC
102	2wRTD85	MAV:3	100:150	2	T:EC
103	2wRTD85	MAV:3	100:150	3	T:EC
105	2wRTD85	MAV:3	100:150	4	T:EC
106	2wRTD85	MAV:3	100:150	5	T:EC
108	2wRTD85	MAV:3	100:150	6	T:EC
109	2wRTD85	MAV:3	100:150	7	T:EC
110	2wRTD85	MAV:3	100:150	8	T:EC
111	2wRTD85	MAV:3	100:150	9	T:EC
112	2wRTD85	MAV:3	100:150	10	T:EC
113	K	MAV:3	100:150	11	T:EC
114	K	MAV:3	100:150	12	T:EC
115	K	MAV:3	100:150	13	T:EC
116	J		100:150	14	T
117	J		100:150	15	T
118	J		100:150	16	T

Kuva 11: HTR:n valvomien lämpömittausten määrittely

Kuvissa 10 ja 11 näkyy Excel-tiedoston osuudet, joissa määritellään tallennettavat parametrit ja lämpömittaukset. Parametrit eritellään niiden laiteohjelmistossa määritellyn parametrinumeron perusteella ja muoto datan tyypin ja koon perusteella. Kaikille parametreille voi asettaa minimi- ja maksimirajat. Parametreille voi myös kirjoittaa selittävän tekstin sekä halutessaan asettaa sarakkeeseen DisplayFormat esitystarkkuuden desimaalien määränä. Actions-sarakkeeseen kirjoitetaan HTR:n ikkunat, joihin parametrit halutaan näkyviksi.

Lämpömittausten määrittely tapahtuu samankaltaisesti kuin parametrienkin. Mittaukset tunnistetaan niiden kanavan numerosta (101 – 120) ja anturin tyypistä. Esimerkkikuvassa on 10 vastuslämpömittaria sekä kolme K- ja J-tyypin termoparia. Lämpömittauksille asetetaan kaksoispisteellä erotettuna varoitus- ja hälytysraja, joka katkaisee testin. Jokaiselle mittaukselle voi myös halutessaan asettaa suodatuksen sarakkeeseen TempFilter.

Haluttu testisykli määritellään XML-tiedostoon, joka sijaitsee myös ohjelman kansiossa. Erilaisille testityypeille on kaikille oma XML-tiedostonsa, johon muutetaan testityypin suorittamiseen tarvittavat parametrit.

```
- <Target Name="ABB_HTR">
  - <Points>
        <!-- Speed sync for the drives -->
    - <Point StepName="Speed sync" TestName="Constant_load Heatrun">
        <tempTolerance>3</tempTolerance>
        <tMaxMinutes>5</tMaxMinutes>
        <tWindowMinutes>5</tWindowMinutes>
        <Speed>980</Speed>
        <Torque>10</Torque>
      </Point>
        <!-- Test 1 - Constant Load heatrun -->
    - <Point StepName="Constant_load Heatrun" TestName="Constant_load Heatrun">
        <tempTolerance>1</tempTolerance>
        <tMaxMinutes>5</tMaxMinutes>
        <tWindowMinutes>5</tWindowMinutes>
        <Speed>980</Speed>
          <!-- NB: Whether 'Speed'(rpm)^ or 'Freq'(Hz)_ will be used (for EUT) depends on EUT CM settings in xls config -->
        <Freq>0</Freq>
        <Torque>20</Torque>
          <!-- Used only if PIDMode is 0 -->
        <PIDMode>0</PIDMode>
          <!-- Current setpoint -->
        <Setpoint>1060</Setpoint>
        <PID_dt>2</PID_dt>
        <Kp>0.1</Kp>
        <Ki>0.003</Ki>
        <Kd>0</Kd>
        <PIDMaxValue>40</PIDMaxValue>
        <PIDMinValue>20</PIDMinValue>
        <PIDStartValue>25</PIDStartValue>
        <SpeedStartLimit>500</SpeedStartLimit>
        <SpeedStopLimit>300</SpeedStopLimit>
```

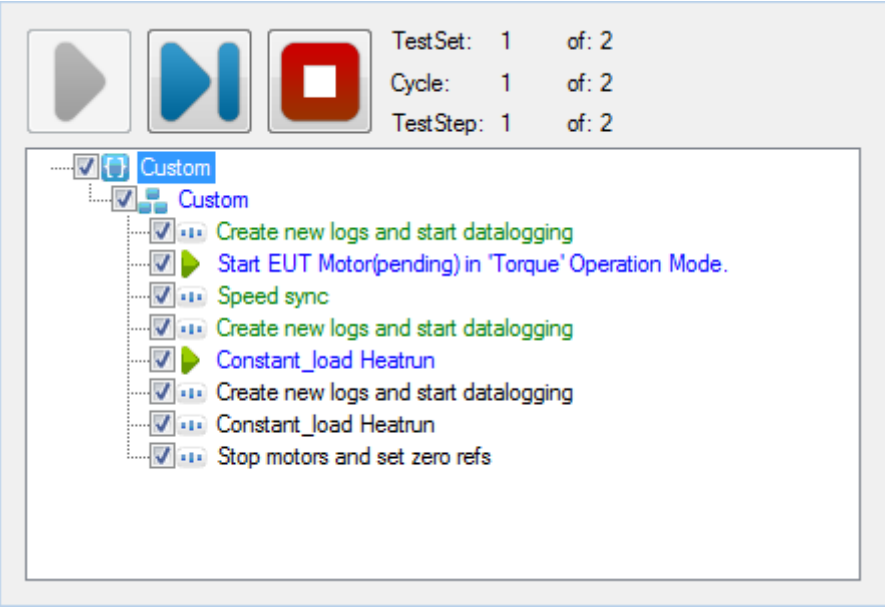
Kuva 12: Esimerkki HTR:n staattisen kuormitustestin määrittelevästä XML-tiedoston osasta

Kuvassa 12 on esitetty osa testipisteen määritelevästä XML-tiedostosta. Kuvassa on kaksi pistettä: testimoottorin ja kuormamoottorin keskinäisen toiminnan varmistaminen Speed sync sekä yksi vakiokuormalla ajettava testipiste. Speed sync -vaiheessa testattavalle taajuusmuuttajalle ja kuormakoneelle annetaan toiselle nopeus- ja toiselle momenttiohje speed ja torque -kohtiin ja taajuusmuuttajat käynnistetään. Pisteen aikana nopeusohjeella ohjattava moottori kiihtyy ja momenttiohjeella ohjattava moottori jarruttaa halutulla, yleensä suhteellisen pienellä momentilla akselia, jolloin varmistutaan siitä, että molemmat taajuusmuuttajat pystyvät mittaamaan testijärjestelmän nopeus- ja momenttipisteen ja säätämään toimintaansa pisteen ylläpitämiseksi.

Kuormitustestipisteessä testilaitteen syöttämä taajuus eli moottorin pyörimisnopeus ja testilinjaston läpi kulkeva teho säädetään halutuiksi. Tehon säätöön voidaan käyttää joko suoraan momenttiohjetta tai HTR:n PID-ohjainta, joka pyrkii pitämään sen ohjaamaksi asetettua muuttujaa vakiona. Virta on yleensä kuormitustesteihin määritelty parametri, jota PID ohjaa ja PID-säätimellä virta saadaan pidettyä tarkemmin tasaisena pitkissä testeissä, joissa jännitetaso saattaa heilahdella verkon kuormituksen mukaan. PID-säädin vaatii kuitenkin moottorikohtaisen säädön, jotta se toimii halutulla tavalla eikä anna virtapiikkejä tai ala värähdellä. Sen käyttö on siis hieman haastavampaa kuin pelkän momenttiohjeen antaminen, mutta tuottaa oikein toimiessaan tarkemman testipisteen.

Testipisteen loppumisehdoksi voidaan asettaa maksimiaika sekä mitattujen lämpötilojen tasaantuminen halutulla tarkkuudella. Testipisteen maksimiaika asetetaan kohtaan tMaxMinutes ja lämpötilojen tasaantumista tarkkaileva ehto asetellaan kohtiin tempTolerance ja tWindowMinutes. Esimerkiksi arvot tMaxMinutes = 60, tempTolerance = 3 ja tWindowMinutes = 5 asettaa testipisteen sellaiseksi, että piste on valmis, kun valitut lämpötilat ovat muuttuneet alle kolme astetta viimeisen viiden minuutin aikana tai pistettä on ajettu yli 60 minuuttia.

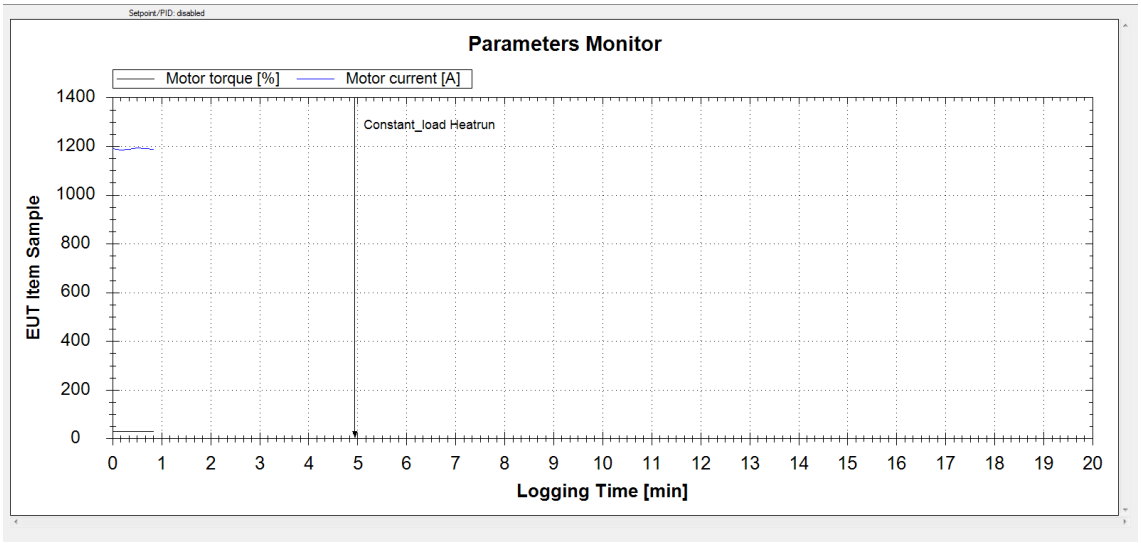
Kun XML-tiedosto on aseteltu, voidaan HTR käynnistää. HTR:n käyttöliittymä koostuu useista ajoikkunoista, joista testin kulkua voi valvoa.



Kuva 13: Testin kulun seuranta ja ohjaus

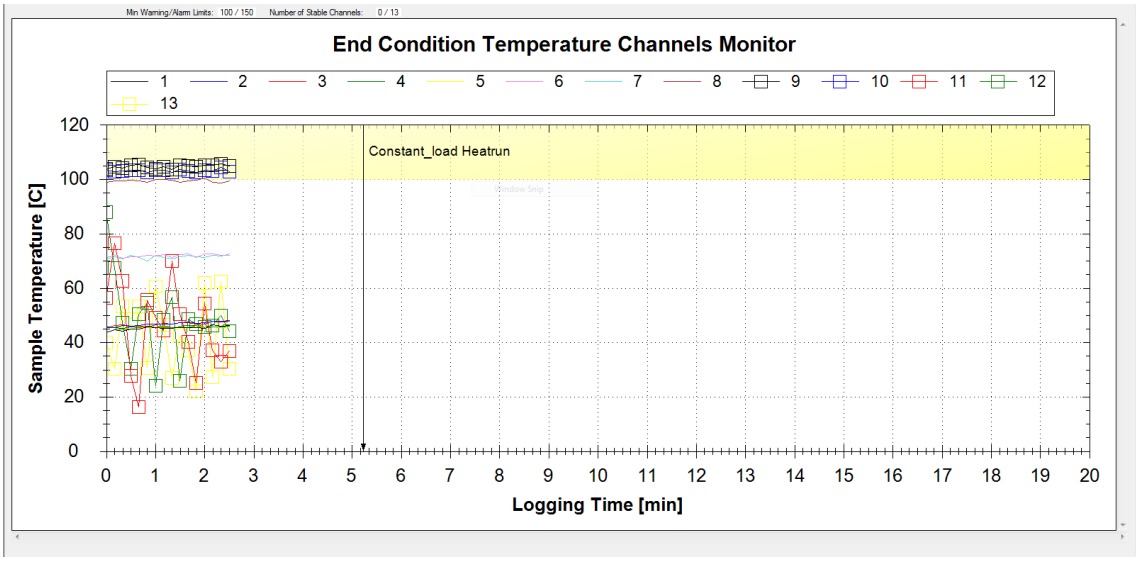
Kuvassa 13 on HTR:n ajoikkuna, josta testaaja näkee asetellut testipisteet ja testin kulun. Ikkunasta pystyy myös käynnistämään ja pysäyttämään testin sekä ohittamaan testin vaiheita haluttaessa. Kuvan esimerkissä on aseteltu kaksi kuormapistettä (Constant_load Heatrun), jonka lisäksi testiin kuuluu käynnistyksen ja sammutuksen toteuttavia vaiheita sekä mittaustulosten keräämisen aloittaminen ja lopettaminen.

Kaikki mitattavat suureet saa testin aikana luettaviksi erilaisille kuvaajille. Taajuusmuuttajan parametrit ja sisäiset muuttujat saa luettavaksi myös numeroarvoina. Eri muuttujien arvot saadaan esitettyä ajan funktiona (kuvat 14 ja 15). Lämpötilavalvonnan saa esitettyä myös suhteessa varoitus- ja hälytysrajaan (kuva 16).



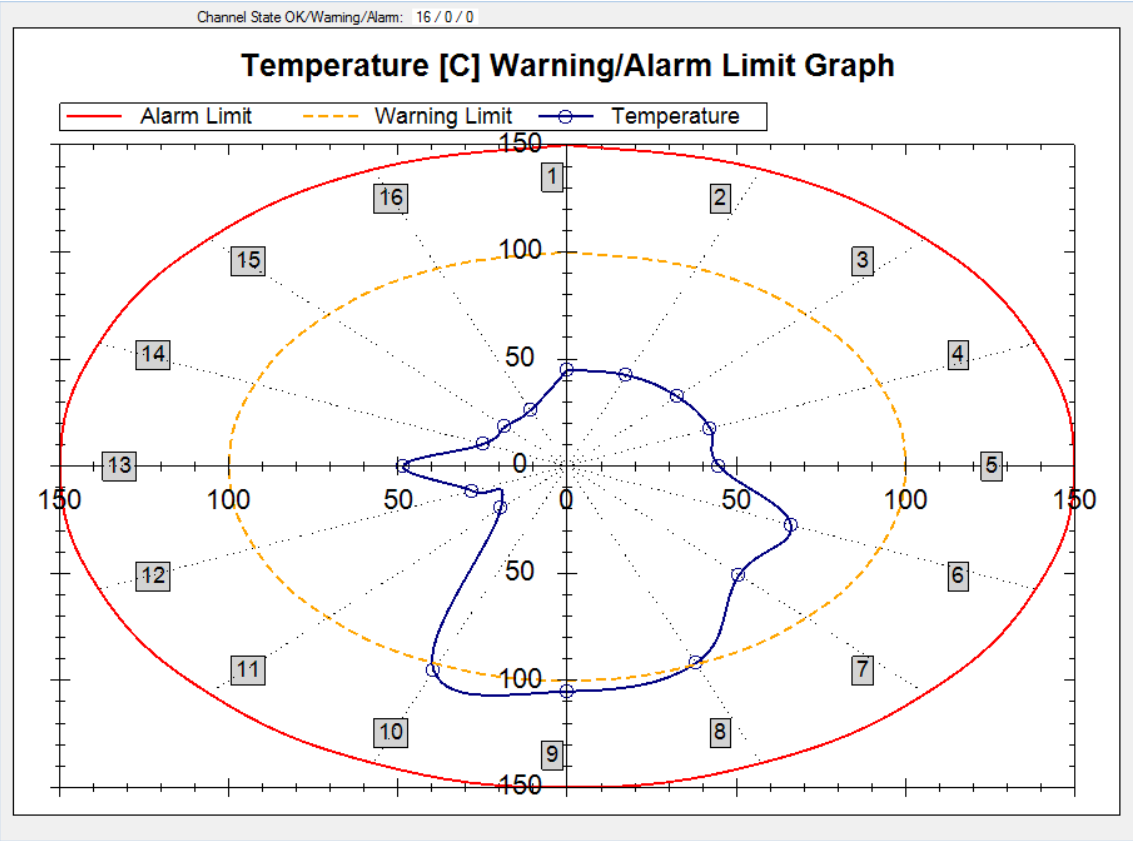
Kuva 14: Valittujen parametrien arvot ajan funktiona

Kuvan 14 kuvaajasta voidaan seurata eri parametreissa tapahtuvia muutoksia testin aikana. Kuvaajan avulla voi seurata odottamattomia muutoksia esimerkiksi testivirrassa ja tarkastaa takautuvasti testin kulkua.



Kuva 15: Testinaikaiset lämpötilat ajan funktiona

Kuvassa 15 esitetystä lämpötilakuvaajasta nähdään, että useiden lämpötilojen tarkkaileminen tässä muodossa saattaa olla haastavaa. Huomioitavan arvoista on myös, että vialliset mittaukset, kuten tässä tapauksessa mittaukset 11 ja 13 jotka värähtelevät huomattavasti, estävät ohjelmaa havaitsemasta lämpötilojen tasaantuneen, mikä on yleensä kuormitustestien lopetusehto. Häiriöinen tai viallinen mittaus saattaa aiheuttaa myös testiajoon ylimääräisiä hälytyksiä ja katkoksia, jos sille on asetettu hälytysraja. Tämän takia mittausten toiminta pitäisi tarkastaa ennen pidempiä testejä ja niitä pitää suodattaa tarvittaessa. Varmistuskeinona testien etenemiselle käytetään testipisteen maksimiaikaa, joka voidaan asettaa kokemuksen perusteella turvallisen pitkäksi, niin että mitatut lämpötilat ovat siihen mennessä todennäköisesti tasaantuneet.



Kuva 16: Mitatut lämpötilat esitettynä suhteessa varoitus- ja hälytysrajaan

Testien valvonnan kannalta Kuvan 16 kuvaaja on kuvan 15 näkymää havainnollisempi tapa seurata kriittisten pisteiden lämpötilaa. Kuvan esimerkkitapauksessa kuvasta nähdään, että mittaukset 8, 9 ja 10 ovat varoitusrajan kohdalla ja muut mittaukset turvallisella alueella. Tästä esitysmuodosta pystyy seuraamaan nopeasti, onko jokin seuratuista lämpötiloista ylittänyt varoitusrajan tai muuten odottamattoman korkea. Kuvan esimerkissä varoitus- ja hälytysrajat on asetettu kaikissa tarkkailluissa kanavissa yhtä suuriksi, mutta jokaiselle kanavalle voi asettaa yksilöllisen mittauksen kohteen lämmönkestävyydestä riippuvan rajoituksen.

Testin aikana HTR tallentaa mittausdatan myös tiedostoon. Testin jälkeen tiedostosta pystyy lukemaan ja keräämään valittujen mittausten hetkellisarvot testin asettelussa määritellyllä näytteenottovälillä.

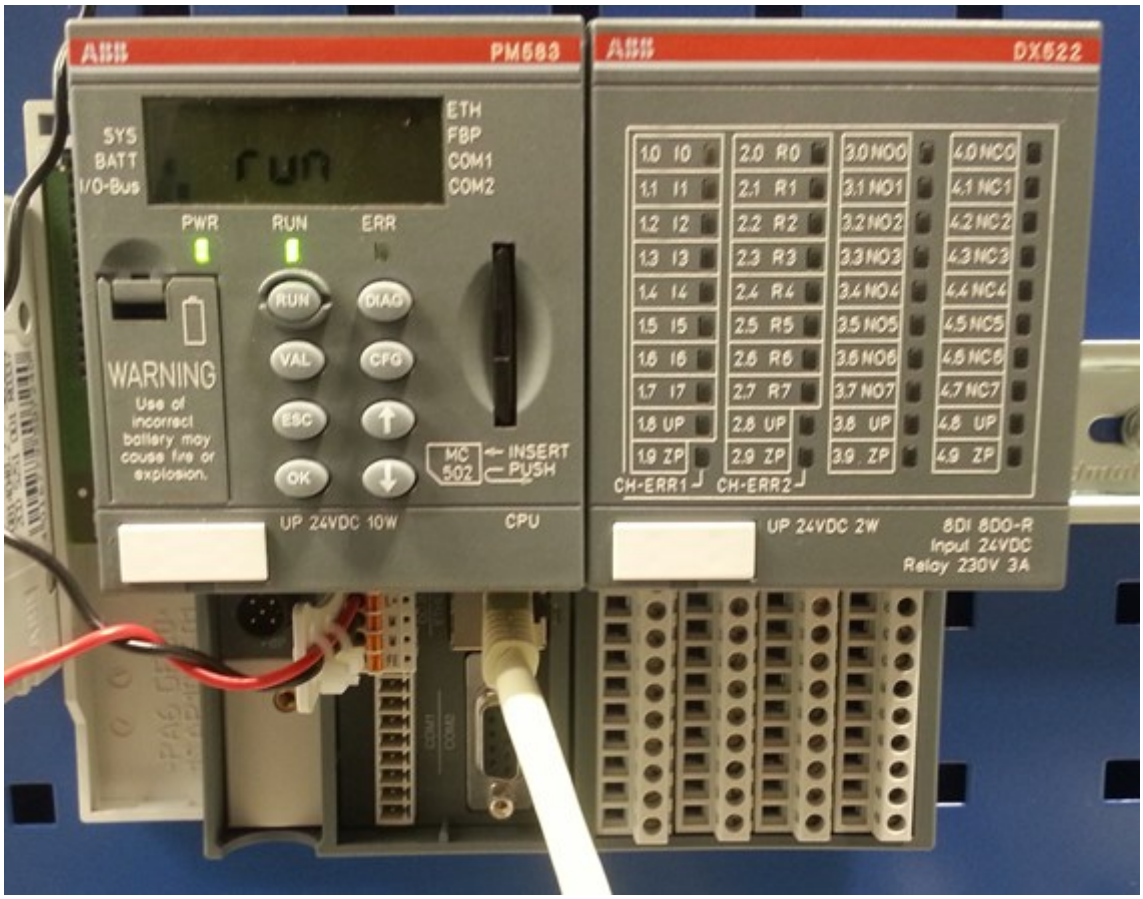
TimeStamp	Elapsed [s]	Motor speed rpm	Motor torque [%]	Motor current	DC voltage [V]	1	2	3	
20140822-173559-1s	0.000	980.420	19.400	980.060	729.110	42.87	45.05	44.28	4
20140822-173609-1	10.001	980.200	19.400	969.880	731.230	43.44	44.56	45.52	4
20140822-173619-1	20.001	980.240	19.400	981.100	727.620	43.48	45.21	44.51	4
20140822-173629-1	30.002	980.250	19.400	980.610	729.120	43.48	45.34	44.89	4
20140822-173639-1	40.002	980.240	19.400	990.280	734.530	43.98	45.45	45.28	4
20140822-173649-1	50.002	980.240	19.400	983.460	727.530	44.21	45.66	45.20	4
20140822-173659-1	60.003	980.230	19.400	973.380	729.530	43.83	45.28	45.63	4
20140822-173709-1	70.002	980.240	19.400	982.250	730.450	44.67	46.09	45.27	4
20140822-173719-1	80.002	980.220	19.400	980.600	730.430	44.59	45.94	45.63	4
20140822-173729-1	90.002	980.270	19.500	987.840	734.840	44.10	45.90	46.83	4
20140822-173739-1	100.003	980.230	19.400	976.610	730.650	44.78	45.83	45.72	4
20140822-173749-1	110.002	980.220	19.400	980.420	731.570	45.08	46.86	46.66	4
20140822-173759-1	120.002	980.230	19.400	987.920	736.420	44.52	46.83	46.26	4

Kuva 17: HTR:n tallentama mittausdata

Kuvan 17 esimerkissä on lyhyt osuus tallennetusta tiedostosta. Esimerkissä näkyy mittauksen ajankohta, aika testin alusta sekä eri mittaukset sarkaimella erotettuna. Tässä tapauksessa mittauksia on kerätty 10 sekunnin välein. Mittaukset voi kerätä lokitiedostosta jälkikäteen jatkokäsittelyä varten manuaalisesti tai esimerkiksi mittaustuloksien keräämiseen ohjelmoidulla Excel- tai Matlab-makrolla. HTR:ään yhdistettynä ei tällaista makroa ole vielä kehitetty.

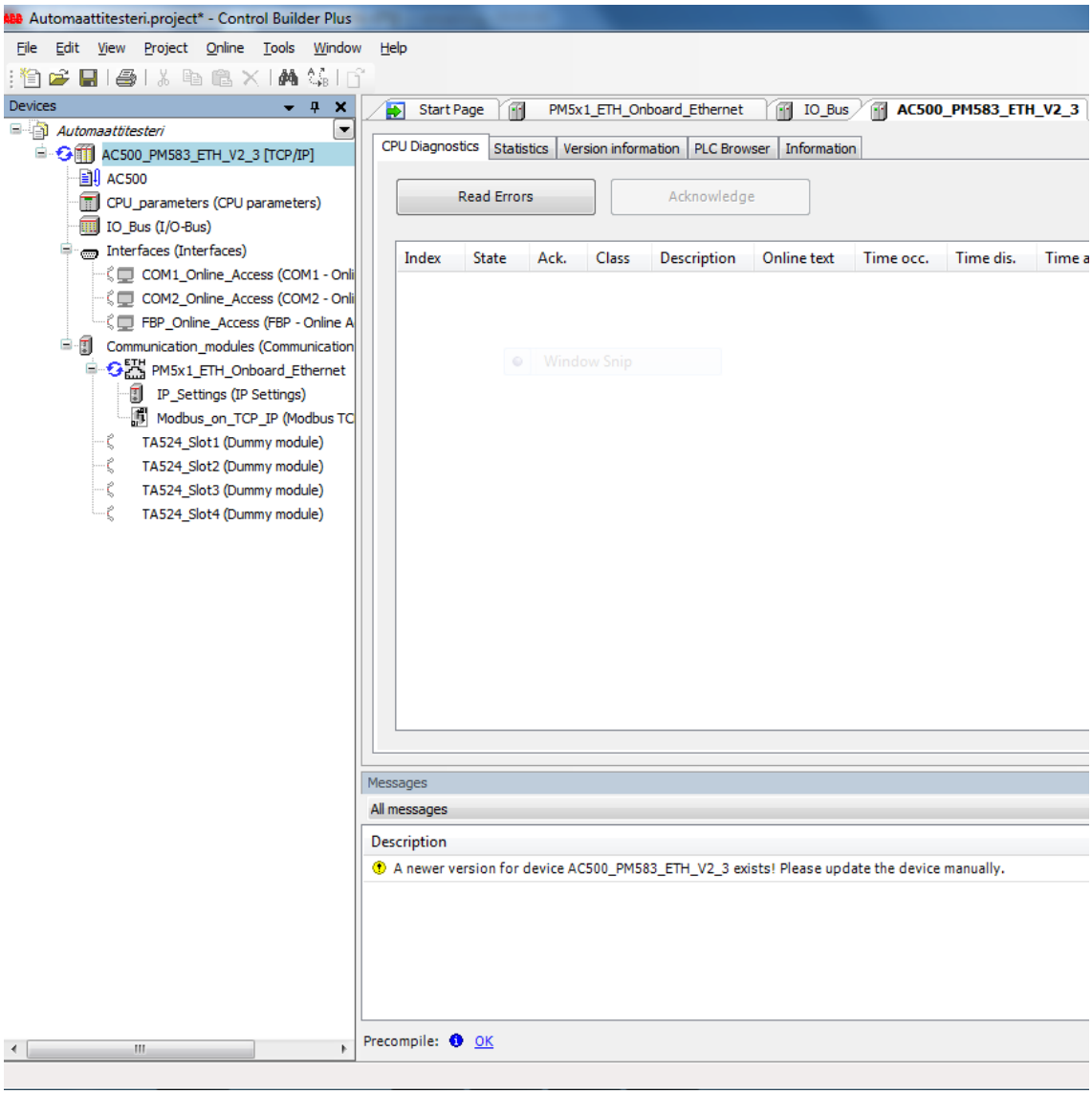
9.2 AC500-ohjelma

AC500 on ABB:n kehittämä kaupallinen ohjelmoitava logiikka. Ohjelmoitava logiikka on tarkoitettu tosiaikaisten automaatioprosessien ohjaamiseen keskitetysti esimerkiksi kenttäväylän yli. AC500:n on liitettävissä myös tarvittaessa digitaalisia ja analogisia sisään- ja ulostuloja laitteiden suoraa ohjausta varten ja useita muita ohjausmahdollisuuksia liitettäväksi logiikkaohjelman käytettäväksi. [15]



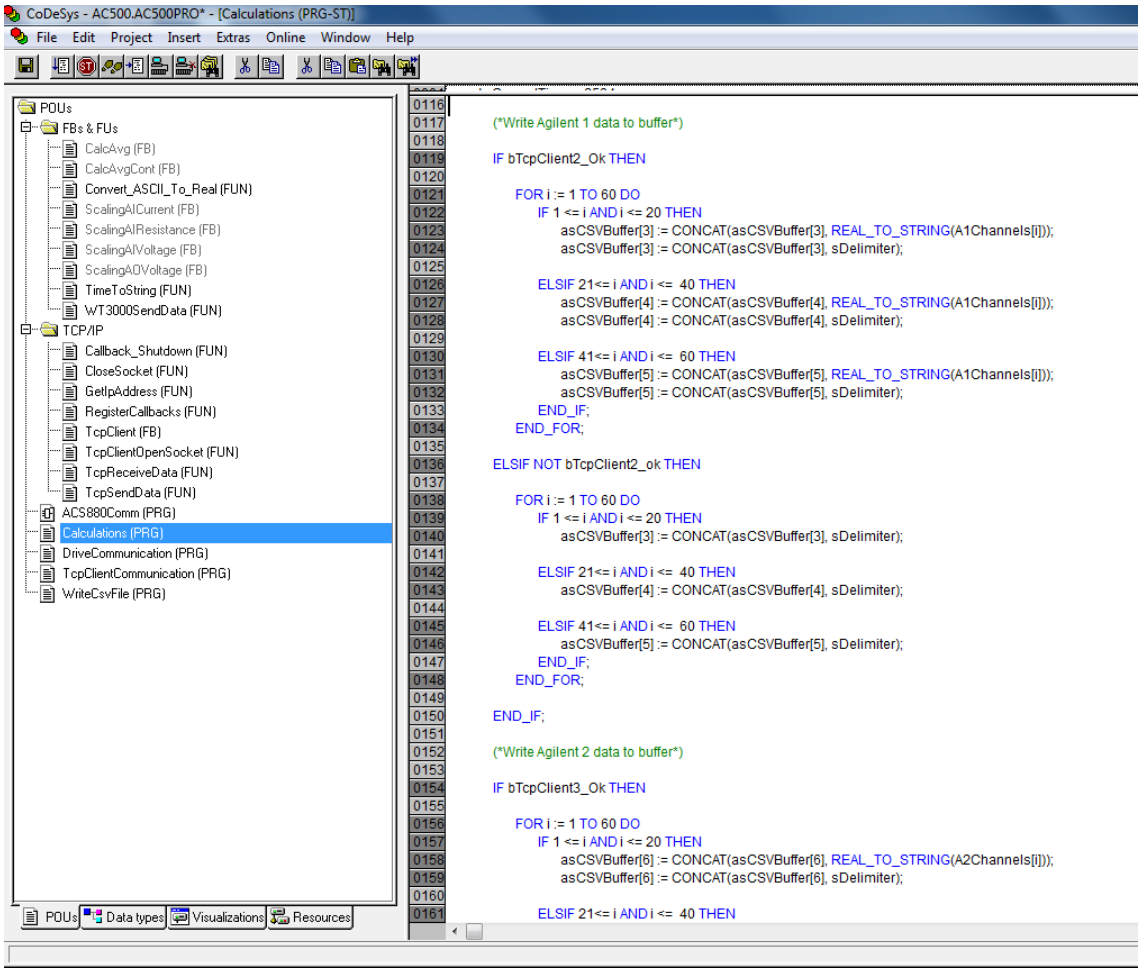
Kuva 18:AC500 PM583 ohjelmoitava logiikka

Tässä diplomityössä kehitetyssä testilaitteistossa käytetään AC500 PM583 -logiikkaa (kuva 18), joka sisältää logiikkaa ohjaavan tietokoneen lisäksi SD-muistikorttipaikan, Ethernet-väylän sekä RS-232-sarjaväylän. Testijärjestelyyn on lisäksi hankittu DX522 releohjausmoduuli, jota ei kuitenkaan ole toistaiseksi otettu käyttöön.



Kuva 19: Control Builder -ohjelman näkymä

AC500-logiikkaa käytetään Windows-koneella Control Builder -ohjelmalla (kuva 19). Control Builderiin määritellään käytetty prosessorityyppi sekä järjestelmään liitetyt kommunikaatio- ja muut moduulit ja niiden osoitteet. Joissain tapauksissa Control Builderissa määritellään myös logiikan ohjaamia laitteita. Asetusten perusteella ohjelma luo Codesys-ohjelmointiympäristöön ohjelmarungon, johon laitteiston käyttämät väylät ja portit on yhdistetty ohjelman muuttujiin. Control Builderin käyttöliittymää voidaan käyttää myös AC500-laitteiston moduulien toiminnan valvontaan logiikkaohjelman ollessa käynnissä. Tällä tavoin käyttäjä pystyy valvomaan prosessorin kuormitusta, virhetiloja sekä eri moduuleiden läpi kulkevaa tietoliikennettä.



Kuva 20: Codesys-ohjelmointiympäristön näkymä

Varsinainen logiikkaohjelma ohjelmoidaan Codesys-ympäristössä (kuva 20). Codesys on teollisten automaatiosovellusten toteuttamiseen kehitetty IEC61131-3 -standardin mukainen ohjelmointiympäristö. Standardi määrittelee kaksi tekstimuotoista ja kolme graafista ohjelmointikieltä. Codesys sisältää myös joukon kirjastoja reaaliaikaisen järjestelmän toimintoja tukemaan. Kirjastot tarjoavat rajapintoja useisiin eri teollisiin tietoliikenne- ja ohjausstandardeihin. [16] Tätä työtä tehdessä käytettiin IEC61131-3:n määrittelemää tekstimuotoista ohjelmointikieltä ST (Structured text) sekä lohkokaaviomuotoista ohjelmointikieltä FBD (Function Block Diagram).

Ohjelmoinnin lisäksi Codesys-ympäristö mahdollistaa logiikkaohjelman reaaliaikaisen käytönaikaisen seuraamisen joko simuloituna tai todelliseen ohjelmoitavaan logiikkaan ladattuna. Ohjelmaa seuratessa käyttäjä voi seurata ja halutessaan asettaa ohjelman muuttujien tiloja ja seurata ohjelman kulkua. [17] Tämä on erityisen hyödyllinen toiminto, kun logiikkaohjelman toteutusvaiheessa etsitään vikoja ohjelman toiminnasta.

Tämän diplomityön logiikkaohjelma ohjelmoitiin ABB Drivesin LAC-osaston automaattitestaukseen kehittämän Codesys-ohjelman pohjalle. Ohjelma on ollut LAC:lla hyötysuhdemittauskäytössä. Testijärjestely on mitannut kiinteästi testilaitteen ja ympäristön lämpötiloja, testilaitteiston akselilla olevaa momenttianturia ja testin sähköisiä suureita yhdellä Agilent 34972A:lla ja kahdella WT3000:lla. Lisäksi ohjelma on ohjannut testiympäristönä toimivan lämpökaapin lämmönsäätöä. Logiikkaohjelma on lähettänyt mittauksensa palvelimella olevaan tietokantaan, josta tuloksia on tarkkailtu.

Valmis ohjelmarunko tarjosi ratkaisun useisiin testijärjestelyn vaatimuksiin. Kommunikoinnin perusta TCP/IP-protokollalla AC500:n ja mittalaitteiden välillä oli ratkaistu valmiiksi. Lisäksi ohjelman runko ja muuttujat oli määritelty LAC:n versiossa suurimmaksi osaksi. Diplomityön aikana ohjelmasta karsittiin toistaiseksi turhia toimintoja, kuten kaapin lämpötilan säätöön ja analogisiin mittauksiin liittyvät funktioryhmät. Lisäksi LAC:n järjestelyssä on käytössä ohjauspaneeli, jolta testin mittauksia ja kaapin säätöjä on voitu seurata ja mittalaitteet kytkeä päälle tai pois. Ohjauspaneelikin on jätetty tämän työn laitteistosta pois. Myös muuta laskentaa karsittiin johtuen siitä, että ne ovat liittyneet suurelta osin mekaanisten ja sähköisten tehojen ja hyötysuhteiden vertailuun. Työssä kehitettävässä testilaitteistossa ei ole käytettävissä momenttianturia ja mekaanisten suureiden mittaaminen ja taajuusmuuttajan toiminnan vertaaminen niihin olisi näin mahdotonta. Diplomityön tavoitteena on käyttöönottaa mahdollisimman geneerinen testausjärjestely, jota voi jatkokehityksenä muokata ja laajentaa helposti.

LAC:n testijärjestelyyn on asennettu kiinteästi kahdeksan lämpömittausta, ja ne on määritelty suoraan ohjelmakoodiin. Tältä osin logiikkaohjelmaa täytyi muuttaa niin, että se tukisi tarvittaessa vaihtelevaa määrää mittauksia. 34972A:n kommunikointia käsittelevään ohjelman osaan määriteltiin kaksi mittalaitetta lisää kiinteisiin IP-osoitteisiin ja kaikille kolmelle laitteelle aseteltiin luettavaksi ja tallennettavaksi 60 mittauskanavaa. Mittalaitekommunikointiin ja mittausten tallentamisen toteuttava ohjelmakoodi toteutettiin ST-ohjelmointikielellä.

Tämänhetkisen ratkaisun ongelma on, että 34972A lähettää mittausdatan pilkun erottamana sarjana. Jos mittaukset on asennettu niin, että väliin jää tyhjiä mittauskanavia, ei ohjelma hyppää näiden paikkojen yli vaan tallentaa yhden mittalaitteen tulokset peräkkäin ja jättää tyhjän tilan mittalaitteen 60 kanavan loppuun. Jokaisen mittalaitteen tulokset on kuitenkin eroteltu toisistaan, koska ne tulevat ohjelmalle eri viesteissä.

WT3000:n kommunikointia ei tarvinnut logiikkaohjelmassa muokata, sillä siinä on alkuperäisessä ohjelmassa käytetty mittalaitteen tarjoamaa valmista mittaustulosten esitysmuotoa, joka sopii myös tämän työn järjestelyyn. Ohjelma tallentaa WT3000:n kaikkien elementtien jännite- ja virtamittauksen sekä WT3000:n niistä laskeman tehon ja verkon vaihe- ja taajuustiedot erillisiin muuttujiin ja tallentaa lisäksi eri elementtien summavirran ja -tehon omiin muuttujiinsa. WT3000:n käytössä täytyy silti varmistua, että mittalaiteen asetuksiin on asetettu sama esitysmuoto kuin logiikkaohjelma olettaa.

Mittaustulokset on LAC:n järjestelmässä lähetetty lähiverkon yli tietokantapalvelimelle, josta sitä on pystynyt tarkkailemaan reaaliajassa erilaisista kuvaajista. Tietokannan käyttö on tätä diplomityötä tehdessä LAC:lla jatkokehityksen alla niin, että mittaukset tallennettaisiin tulevaisuudessa tietokannassa myös valmiiseen raporttipohjaan. Tässä HPD:n ensimmäisessä automaattitestilaitteen versiossa käytettiin palvelinyhteyden sijasta AC500:n muistikorttia kaiken mittausdatan tallennuspaikaksi. LAC:n kehittämässä logiikkaohjelmassa ohjelma tallentaa SD-kortille pelkästään yleistietoa testijärjestelystä, mutta muistikorttiin kirjoittavaa aliohjelmaa voitiin käyttää mittaustulosten tallennuksen pohjana, ainoastaan mittaustulosten tallentaminen kirjoituspuskuriin ja uuden kirjoituspuskurin tallentaminen täytyi ohjelmoida uudestaan, jotta tallennetun tiedoston muoto on halutun kaltainen.

Tiedostoon kirjoitus on toteutettu niin, että mittalaitteilta kerätyt tulokset tallennetaan kolmen 34972A:n tapauksessa staattisiin liukulukutaulukoihin ja WT3000:n tapauksessa kiinteään liukulukutietueeseen mittalaitteen ulostulon mukaisesti. Taulukoista ja

tietueesta mittaustulokset kirjoitetaan merkkijonona kirjoituspuskurina toimivaan taulukkoon, jossa on 11 alkiota joissa kaikissa on 255 merkkiä. Puskuri on toteutettu tällä tavoin osissa, sillä Codesysin tarjoamien tiedostoon kirjoittamisen toteuttavien kirjastojen havaittiin tukevan maksimissaan 255 merkkiä.

```
⊟asCSVBuffer
  asCSVBuffer[1] = '17:59:11;513.313;304.34;511.395;305.03;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;'
  asCSVBuffer[2] = '0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;'
  asCSVBuffer[3] = '44.146;45.911;45.588;45.588;45.617;71.896;71.061;99.381;103.922;102.495;31.932;46.672;54.682;27.57;27.841;28.117;0.0;0.0;0.0;0.0;'
  asCSVBuffer[4] = '0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;'
  asCSVBuffer[5] = '0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;'
  asCSVBuffer[6] = '25.951;28.239;28.236;28.931;27.962;28.522;28.889;28.684;28.608;29.273;42.111;33.401;30.475;31.916;28.132;26.809;26.681;27.806;27.425;28.763;'
  asCSVBuffer[7] = '26.423;33.969;31.317;29.469;32.755;38.548;33.123;33.09;34.108;34.537;33.159;33.027;36.599;36.375;72.084;31.313;31.371;31.798;28.337;100.748;'
  asCSVBuffer[8] = '0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;0.0;'
  asCSVBuffer[9] = ';;;;;;;;;;;;;;;;;;;;'
  asCSVBuffer[10] = ';;;;;;;;;;;;;;;;;;;;'
  asCSVBuffer[11] = ';;;;;;;;;;;;;;;;;;;;$R$N'
```

Kuva 21: Puskuri, johon tiedostoon kirjoitettavat mittaustulokset kerätään

Kuvassa 21 on esitetty tallennuspuskuri asCSVbuffer. Puskurin kaksi ensimmäistä alkiota on varattu mittauksen ajankohdalle ja WT3000:n mittauksille ja yhdeksään muuhun puskuriin tallennetaan jokaiseen yhden 34972A:n lämpömittaukset. Mittaustulokset on eroteltu puolipisteellä ja kirjoitetaan kaikki samalle riville. Kuvan esimerkissä WT3000:lla on kahdessa elementissä virta- ja jännitemittaus, mutta ei laskennallisia arvoja ja käytössä on kaksi 34972A-tiedonkeruulaitetta, joissa ensimmäisessä on yksi ja toisessa kaksi multiplekserimoduulia. Tyhjän mittalaitteen tilalle kirjoitetaan pelkkää puolipistettä, jotta tallennetun tiedoston sarakkeet pysyvät oikeilla paikoilla.

Puskurista mittaustulokset tallennetaan 10 sekunnin välein AC500:n muistikortille. Testin alkaminen aloittaa uuden tiedoston, johon merkitään testin numero, päivä ja alkamisaika sekä mittausten otsikkotiedot. Mittaustulokset tallennetaan sarakkeittain ja jokainen tallennettu mittaushetki allekkain.

Test number	15							
Test start date	22.8.2014							
Test start time	17:59:01							
Time	E1 U	E1 I	E2 U	E2 I	E3 U	E3 I	E4 U	E4 I
17:59:11	513.313	304.34	511.395	305.03	0.0	0.0	0.0	0.0
17:59:21	513.979	303.52	512.353	304.11	0.0	0.0	0.0	0.0
17:59:31	514.096	303.9	512.942	303.98	0.0	0.0	0.0	0.0
17:59:41	514.814	303.99	513.578	303.26	0.0	0.0	0.0	0.0

Kuva 22: Tiedosto, johon mittaustulokset tallennetaan, I osa

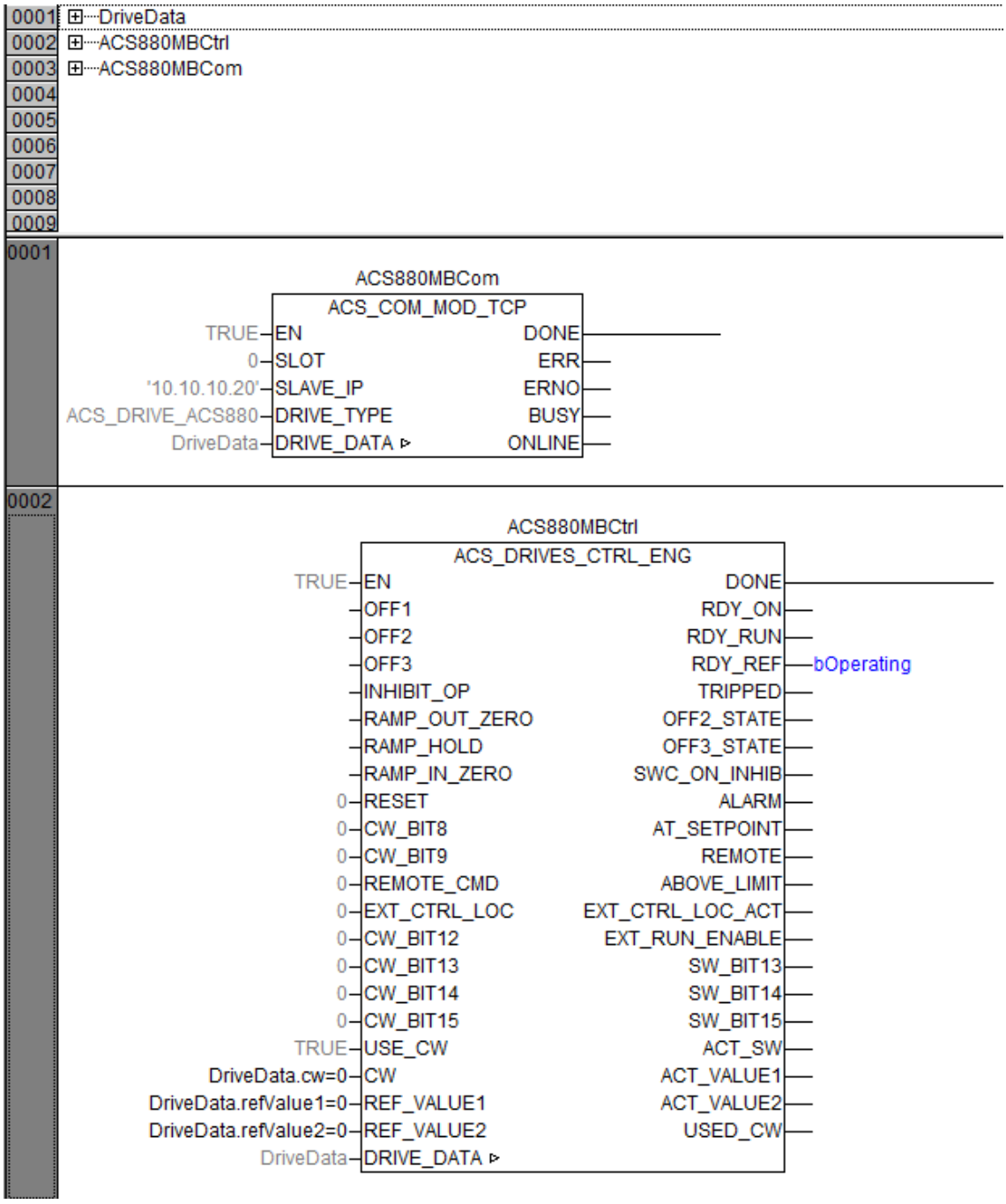
A1:101	A1:102	A1:103	A1:104	A1:105	A1:106	A1:107	A1:108	A1:109
44.146	45.911	45.588	45.588	45.617	71.896	71.061	99.381	103.922
44.506	46.243	46.3	45.534	45.965	71.478	70.904	98.828	103.499
44.229	46.62	46.189	45.77	46.09	71.766	70.487	98.381	103.922
44.783	46.176	45.646	45.213	45.825	71.348	69.652	98.828	104.926
44.77	45.646	46.425	45.924	46.077	71.478	70.643	99.25	104.926
45.117	46.508	46.16	46.105	45.492	71.766	70.904	99.513	103.79
43.965	45.965	46.425	45.742	45.812	71.191	70.487	98.671	103.922
44.465	45.924	45.269	45.088	45.783	71.061	70.904	99.092	104.477

Kuva 23: Tiedosto, johon mittaustulokset tallennetaan, II osa

Kuvissa 22 ja 23 on osa tiedostosta, johon mittaustulokset tallennetaan. Kuvan 22 osiossa on testin otsikkotiedot ja mittaustulosten alkupää. Sarakkeeseen Time kirjoitetaan mittauksen ajankohta. Sarakkeisiin E1 – E4 U ja E1 – E4 I kirjoitetaan WT3000:n elementtien jännite- ja virtamittaukset. Kuvassa 23 on esitetty pieni osa lämpömittauksien sarakkeista. Mittaukset on eritelty mittalaitteen tunnuksella A1 – A3 sekä kanavan tunnuksella 101 – 320, missä ensimmäinen numero kuvaa mittauksen moduulia ja kaksi viimeistä kanavan paikkaa.

Mittalaitekommunikaation lisäksi testilaitteiston täytyy kommunikoida taajuusmuuttajan kanssa, jotta laitteisto saa tiedon testin alkamisesta ja päättymisestä. Tieto otetaan taajuusmuuttajan tilasanan bitistä, joka kertoo moduloiko taajuusmuuttaja vai ei. Testin todetaan olevan käynnissä, jos bitin arvo on 1. LAC:n testijärjestelyssä taajuusmuuttajan ja AC500:n kommunikaatio on toteutettu Profibus-väylällä, joka vaatii erillisen kaapelin laitteiden välille. Tämä kommunikaatio muutettiin tässä työssä Modbus/TCP-protokollalla toteutettavaksi, sillä protokolla käyttää signaalitienä helposti saatavissa olevaa RJ45-kaapelia ja AC500 sekä ABB:n taajuusmuuttajat tarjoavat Modbus-kommunikointiin valmiin rajapinnan. Näin koko testilaitteiston kommunikaatio saatiin toteutettua Ethernet-verkolla.

Taajuusmuuttajan ja AC500:n välinen kommunikointi ohjelmoitiin FBD-ohjelmointikielellä, jossa funktiot on ilmaistu lohkokaavioina ja ohjelman toteutus voidaan kirjoittaa suoraan lohkon sisään- ja ulostuloja käyttämällä.



Kuva 24: Taajuusmuuttajan ja AC500:n välisen kommunikaation toteutus

Kommunikaation toteutus on esitetty kuvassa 24. Modbus-kommunikaatioon käytettiin funktioryhmää ACS_COM_MOD_TCP, joka toteuttaa Modbus-protokollan mukaisen viestinvälityksen ABB Drives Profile -ohjaustavan avulla. ABB Drives Profile välittää taajuusmuuttajan tilan ACS_DRIVE_DATA_TYPE-tyypin tietorakenteessa, joka sisältää taajuusmuuttajan tunnistetiedot, tilasanan, ohjaussanan, kaksi valittua tilatietoa ja kaksi valittua ohjausparametria. Tilasanasta logiikkaohjelma pystyy lukemaan taajuusmuuttajan olotilan ja mahdolliset käytön estot ja ohjaussanaan voidaan puolestaan ohjelmassa kirjoittaa käynnistys-, sammutus- ja muita käskyjä. Tilatietoja ja ohjausparametreja käytetään normaalitilanteessa nopeus-, momentti- tai virtatiedon lukemiseen ja ohjearvojen muuttamiseen. [17]

Lisäksi työssä käytettiin apuna funktioryhmää ACS_DRIVE_CTRL_ENG, joka purkaa taajuusmuuttajan lähettämän tietorakenteen muuttujat erillisiksi ulostuloiksi ja esittää ohjaus- ja tilasanojen bitit eriteltyinä [17]. Näin käyttäjä voi ohjelmassaan muokata suoraan haluamaansa bittiä tai ohjetta, eikä ohjelmaan tarvitse erikseen suunnitella tietorakenteen purkavaa osiota. Tällä hetkellä funktioryhmää käytetään tämän diplomityön ohjelmassa vain yhden tilabitin lukemiseen, mihin kommunikoinnin toteuttava aliohjelma on varsin järeä ratkaisu. Ajatuksena on kuitenkin mahdollisuus laajentaa taajuusmuuttajakommunikaatiota helposti nykyisen rakenteen päälle.

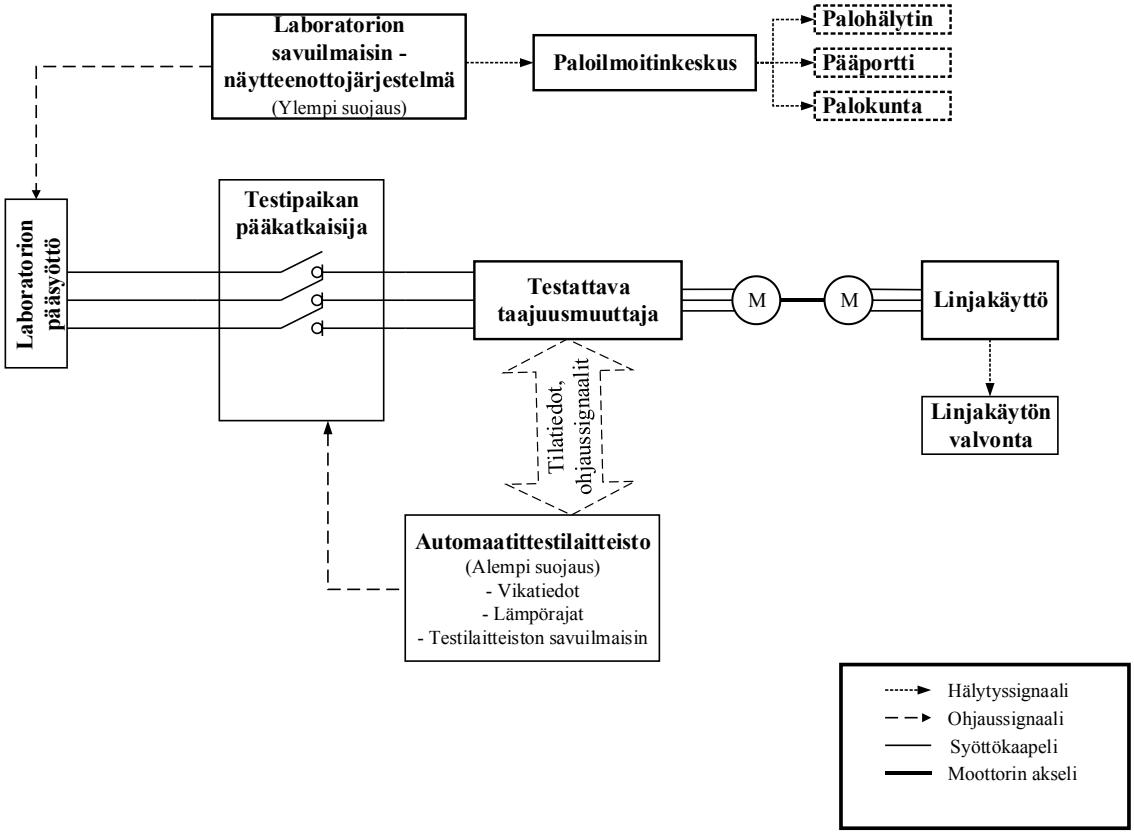
9.3 Turvajärjestelyt

Automaattisen testilaitteiston käytössä tulee ottaa huomioon myös järjestelyn turvallisuus. HTR valvoo taajuusmuuttajan toimintaa ja lämpötiloja, mutta tästä huolimatta täytyy varautua odottamattomiin tilanteisiin testin aikana. Testauksen turvallisuuteen pitää kiinnittää erityistä huomiota, kun kyseessä on valvomaton testi.

Tärkeä turvallisuutta varmistava tekijä on itsenäiseen testaukseen soveltuvien laitteiden valinta. Prototyyppilaitteen ensimmäisiä käynnistyskertoja ei voi suorittaa valvomatta, ja tämän takia pitääkin päättää tietty toimintavarmuuden taso, jolloin taajuusmuuttajan voi antaa olla käynnissä ilman välitöntä valvontaa. Sopivia ehtoja ovat esimerkiksi tietty koestusrutiini asennuksen jälkeen, tietty määrä testattuja tunteja laitteella ennen itsenäistä ajoa ja ettei testilaitteeseen ole vastikään tehty komponentti- tai suunnittelumuutoksia joiden toimintaa ei ole testattu kattavasti.

Palotilanteessa laboratorion savuilmaisimet laukaisevat hätäseis-piirin ja antavat hälytyksen palokunnalle sekä tehtaan vartijoille, joiden valvomo sijaitsee tehtaan pääportilla. Palohälytys aiheuttaa paljon lisätyötä, ja jos kyseessä on todellinen tulipalo eikä kuumentumisen aiheuttama savuaminen, aiheuttaa se myös vaaratilanteen. Tämän takia on toivottavaa, että viallinen testi saadaan pysäytettyä ennen paloa. Valvomatta käynnissä olevien käyttöjen yhteyteen liitettäväksi on suunniteltu ylimääräistä savuilmaisintasoa, joka katkaisee pelkästään kyseisen testipaikan syötön heti, kun vähänkin savua alkaa ilmaantua. Tämä saattaisi pysäyttää testin niin, ettei siitä ehdi aiheutua todellista paloa, vaan pelkästään joidenkin komponenttien liiallista kuumenemista, johon pystytään testin tuloksia tutkiessa puuttumaan. Savuilmaisimeksi on alustavasti suunniteltu Stratos Micra 25 -näytteenottosavuilmaisinta, johon saa asennettua näytteenottoputket koko testattavan taajuusmuuttajakaapin poistoilma-aukoista. Savuilmaisin analysoi näytteenottoputkien kautta tulevan ilman partikkelipitoisuuden muutoksia ja katkaisee relelähtönsä pitoisuuden ylittäessä sallitun rajan. Relelähtö voitaisiin kytkeä joko ohjaamaan testilaitteen sammuksiin tai katkaisemaan testipaikan jännitteet ohjaamalla pääkatkaisija auki. Jos automaattitestauksen käyttö yleistyy, olisi testipaikkojen syöttökenttiin hyödyllistä asentaa erikseen helposti tavoitettava piste, johon releohjauksen voi liittää halutessa.

Turvajärjestelyissä voi myös halutessa hyödyntää AC500:n relelähtöjä, joita voidaan ohjata taajuusmuuttajan tilatietojen perusteella. Lähdöillä voitaisiin toteuttaa esimerkiksi testipaikan sähköjen katkaisu testin pysähtyessä, jotta testattavan muuttajan syöttö ei jää testin loputtua jännitteiseksi.



Kuva 25: Automaattitestijärjestelyn suunniteltu paloturvallisuusratkaisu

Kuvassa 25 on esitetty automaattitestijärjestelyyn suunniteltu paloturvallisuusratkaisu. Tavoitteena on, että laboratoriotason palohälytys laukeaisi vain, kun palokunnan paikalle tulo todella on tarpeellista. Testilaitteiston vikarajojen tai viimeistään savuilmaisimen pitäisi katkaista testi ennen kuin laite todellisuudessa syttyy, jolloin testilaitteen ylikuumat komponentit jäähtyvät turvallisesti. Uusia testipaikkoja suunniteltaessa voi olla aiheellista pohtia tapoja eristää suuretkin testilaitteet niin, että niiden mahdolliset palokaasut ohjautuvat hallitusti yhteen pisteeseen, jonka perusteella voidaan katkaista vielä varmemmin pelkästään yhden testipaikan toimintaa sen vikaantuessa ja palohälytystilanteessa selvittää jo hälytyksen aiheuttaneen ilmaisimen perusteella tarkalleen testipaikka, joka hälytyksen on aiheuttanut.

Palohälytystilanteessa hälytysjärjestelmä kutsuu paikalle palokunnan. Jos valvomaton testilaite syttyy palamaan, pitäisi palokunta pystyä ohjeistamaan paikalle ja varmistaa sammutustyön ongelmaton onnistuminen. Esimerkiksi yöaikaan voi olla tilanteita, ettei tehtaalla ole paikan päällä yhtäkään testipaikasta vastaavaa henkilöä vaan pelkästään vartijoita, joilla ei välttämättä ole tietoa palon alkuperästä muuten kuin palon sijainti laboratoriotasolla. Vaikka palotilanteet ovat toistaiseksi olleet erittäin harvinaisia, lisää automaattitestaus laitteiden testausaikaa ja nostaa myös paloriskiä. Tämän takia palotilanteeseen pitää varautua myös pohtimalla kuinka todellinen tulipalo saadaan sammutettua niin, ettei siitä aiheudu vahinkoa ympäristölle.

Vartijoiden ohjeistus on yksi ratkaisu palotilanteen hallintaan. Vartijoille voidaan ilmoittaa iltapäivisin, mitkä laitteet milläkin paikoilla jäävät yöksi käyntiin, minkä perusteella he pystyvät hälytystilanteessa ohjaamaan palokuntaa oikealle paikalle nopeasti. Myös jonkinlainen toimintaohje palon sammutukseen ja palon jälkeiseen valvontaan voisi olla tässä tapauksessa tarpeellinen. Ongelmana tässä suunnitelmassa on se, että vartiohenkilökunnalla on jo ennestään työtehtävä kiinteistön vartioinnissa, ja työtehtävän laajentamiseen ei välttämättä riitä resursseja.

Toinen vaihtoehto palotilanteiden varalle on päivystyksen järjestäminen. Paikan päällä oleva päivystys pystyisi vastaanottamaan palokunnan tarvittaessa. Uuden työvuoron lisääminen vähentäisi kuitenkin automaattitestauksesta saatua hyötyä ja olisi järkevä ratkaisu vain, jos testilaitteita olisi käynnissä useita, jolloin keskitetyn paikalla olevan valvonnan voisi perustella testiin käytettävien resurssien säästönä. Lähipäivystys on hyvä ratkaisu, jos tehtaalla on samanaikaisesti testaushenkilökuntaa tekemässä muita työtehtäviä kuin pelkästään testinvalvontaa. Tällainen mahdollisuus saattaa olla, jos tuotannon laitteiden koestusta tehdään useassa vuorossa. Päivystyksen lisäämisessä koestuksen tehtävänkuvaan on kuitenkin ongelmallinen samasta syystä kuin vartijoillekin. Tähän ei välttämättä ole aikaa eikä henkilöstöä.

Etäpäivystys palotilanteen varalta on melko hyödytöntä. Etäpäivystäjällä kestää todennäköisesti niin pitkään päästä paikan päälle, ettei hänestä hälytystilanteessa ole hyötyä. Etäpäivystystä voi käyttää lähinnä ilmoitusmuotona testausorganisaatiolle itsenäisesti suoritetun testin ongelmista, jotta niihin on varauduttu ennen seuraavaa työpäivää. Todella pitkäkestoisissa ongelmatilanteissa, jotka vaativat jälkiselvitystä, voi olla hyötyä siitä, että tuotekehityksen testauksen puolesta päivystävä henkilö saapuu paikalle hälytystilanteen jälkeen selvittämään aiheutuneita ongelmia.

10 Työn tulokset ja kehityssuuntia

Tällä diplomityöllä on kaksi tavoitetta: Kehittää automaattitestilaitteisto, jolla pystytään toteuttamaan osa nykyisestä taajuusmuuttajan kuormitustestauksesta ja vapauttamaan näin resursseja järjestelmätestauksen kehittämiseen, ja antaa pohjustusta järjestelmätestauksen kattavuuden parantamiselle esittämällä mahdollisesti toteutettavia testausvaihtoehtoja, joita voidaan työn jälkeen alkaa kehittää mahdollisuuksien mukaan. Tässä luvussa käydään läpi työssä aikaansaadut tulokset ja pohditaan jatkokehityksen mahdollisuuksia. Automaattitestilaitteiston ja testauksen kehittämisen tulokset esitetään erillisissä aliluvuissa.

Tässä työssä kehitetty automaattitestilaitteisto on vain ensimmäinen askel testiautomaation kehittämisessä. Laitteisto toimii, mutta siinä on paljon kehitettävää jotta se olisi niin hyvä kuin mahdollinen. Aliluvussa 10.1 käsitellään työssä toteutetun automaattitestilaitteiston käyttöönottoa ja sille mahdollisia kehityssuuntia.

Järjestelmätestauksen kattavuuden kehittämiseen on tarjolla useita mahdollisia suuntia. Testauksen kehittäminen on jatkuva prosessi ja tässä työssä on esillä vain muutama lähtökohta, joilla testauksen laatua voidaan parantaa. Tämän diplomityön ensimmäisissä luvuissa tutkittiin järjestelmän suunnittelua ja testausta sekä järjestelmätestauksen roolia testauksessa. Tämän jälkeen selvitettiin taajuusmuuttajan ominaisuuksia järjestelmänä ja testauksen toteuttamista HPD:lla. Aliluvussa 10.2 pohditaan tutkitun tiedon perusteella mahdollisia tapoja käyttää automaattitestauksella saavutettua hyötyä taajuusmuuttajan järjestelmätestauksen kehittämiseen.

10.1 Automaattitestilaitteisto

Luvussa 9 esiteltiin tässä diplomityössä kehitetyn automaattitestilaitteiston toimintaperiaate. Suunnittelun ja toteutuksen jälkeen testilaitteiston toimintaa kokeiltiin erilaisissa käyttötilanteissa. Testauksen aikana huomattiin virheitä lähinnä logiikkaohjelman toiminnassa. Tämä oli odotettavaa, sillä AC500 ja Codesys-ohjelmointi opeteltiin tämän työn puitteissa. Ohjelmointivirheitä korjattiin toteuttamalla ohjelmaa osa kerrallaan ja testaamalla osan toiminta. Kun ohjelman eri osat toteuttivat suunnitellun toimintansa, aloitettiin koko testijärjestelyn testaaminen osissa.

AC500:lla toteutettu mittalaitedatan tallentaminen testattiin toimivaksi simuloidulla mittalaitedatalla ja todellisilla mittalaitteilla. Mittalaitedatan havaittiin tallentuvan muistikortille suunniteltuun tapaan molemmissa tapauksissa. Agilent 34972A:n tulokset tallentuivat logiikkaohjelmaan ongelmitta, kun mittalaitteiden IP-osoitteet ja mittaukset oli ensin aseteltu tietokoneella halutuksi. Puutteeksi mittalaitekommunikaatiossa havaittiin, että näytteenottoväli asetellaan ohjelmaan tällä hetkellä ohjelmoituja laskureita tai ohjelmakierron pituutta muokkaamalla. Näytteenottovälin muokkaaminen täytyy halutessa suorittaa useisiin eri ohjelman osiin erikseen, minkä takia se on melko haastavaa.

WT3000:n mittauksia testatessa havaittiin, että suunnittelun mukainen asettelu on logiikkaohjelman toiminnan kannalta oleellista. Logiikkaohjelman käyttämästä mittaustulosten asettelusta poikkeaminen siirtää tämänhetkisessä ohjelmassa mittaustulokset täysin vääriin kohtiin WT3000:n lähettämässä viestissä ja aiheuttaa tulosten tallentumisen väärään kohtaan tai jopa tyhjän tuloksen tallentamisen. WT3000:n

käyttäminen ohjelmoidusta mittausjärjestyksestä poikkeavasti vaatii käytännössä logiikkaohjelman lähdekoodin muokkaamista, mikä on vaivalloista.

HTR käyttöönotettiin ja sillä testattiin taajuusmuuttajan lyhytaikaista ohjausta ja testipisteiden vaihtumista automaattisesti. Lyhyet testit onnistuivat käyttöönoton vaikeuksien jälkeen odotetusti, ja yhdellä testatuista käytöistä virtaohjeen ohjaaminen PID-säätimellä toimi erittäin hyvin. Toisella käytöllä, jossa testattiin suhteellisen pieniä virtaohjeita tiukoilla HTR:lle asetelluilla virtarajoilla, PID-säätimen käyttö aiheutti kuormituksen jatkuvan nousemisen ja laitteen toiminnan katkeamisen ylivirtavikaan. Tämä ei ole missään tapauksessa hyväksyttävää, joten PID-säätimen käyttöön on tapauskohtaisesti perehdyttävä tarkemmin. Alustavasti kokeiltuna sen käyttö moottorikohtaisesti oikein aseteltuna tuottaa aiemmin testauksessa käytettyä manuaalista ohjausta paljon tarkemman testipisteen asettelun. Momenttiohjeen määrittely testipisteisiin käsin, mikä vastaa pääpiirteittäin aiempaa testipisteen asettelua, toimi testeissä ongelmitta.

HTR:n ja AC500:n yhtäaikaista toimintaa kokeiltiin automaattitestilaitteiston testauksen lopuksi. HTR ohjasi testiä ongelmitta ja mittasi yhdellä 34972A:lla lämpötiloja USB-piuhan kautta samalla, kun AC500 oli kytkeytyneenä mittalaitteeseen Ethernet-verkon välityksellä. Ajoittain kahden eri mittaussijainnin yhtaikaiset pyynnöt aiheuttivat mittalaitteeseen vian, joka ei kuitenkaan keskeyttänyt mittausta eikä vaikuttanut mittaustulosten laatuun. Todennäköisesti kyseinen vika aiheuttaa yhden mittauspisteen korvautumisen edellisen tuloksen kopiolla toisen mittauslähteen mittaustuloksista. Tätä toimintoa voi jatkossa kehittää.

AC500 sai testissä taajuusmuuttajan tilasanasta testinaloitusohjeen, mutta HTR:n testipisteen vaihto ei sisällä ainakaan kokeilluilla ohjaustavoilla taajuusmuuttajan pysäyttämistä vaan vaihtaa pisteen lennosta, minkä takia AC500 ei vaihda tallennustiedostoa eri testipisteiden välillä. Tämän takia monen peräkkäin otetun mittauspisteen tulkitseminen täytyisi nykyisellään parsia yhdestä tiedostosta, mikä on työlästä. Joko HTR:n testisekvenssi täytyy asetella sellaiseksi, että se pysäyttää taajuusmuuttajan pisteen vaihtuessa tai AC500:n testitiedostonaloituksen ehto täytyy muuttaa sopivammaksi. Nykyisellä ohjelmalla kiertotie testien tulkintaan lienee kuormittamattoman pisteen määrittely jokaisen testipisteen väliin, minkä perusteella alhaisemmasta virta-arvosta ja lämpötilojen huomattavasta laskemisesta voidaan tulkita testipisteen vaihtuneen. Toiminto vaatii joka tapauksessa kehittämistä.

Kaiken kaikkiaan diplomityössä kehitetty automaattitestilaitteisto vaatii tällä hetkellä paljon asettelua ennen käyttöä, ja jokaisen testin toiminnan varmistukseksi testisekvenssi ja mittausten tallentuminen kannattaa testata lyhennettynä ennen varsinaista automaattitestausta. Käyttöönoton jälkeen testijärjestely kuitenkin toteuttaa staattiset ja sykliset kuormitustestit itsenäisesti.

Seuraavana automaattitestausta kehittävänä askeleena kehitetty testilaitteisto täytyisi tämän diplomityön jälkeen ottaa kiinteäluontoisempaan käyttöön jollekin testipaikalle. Tälle paikalle pitäisi asentaa turvallisuuteen liittyvät ulkoiset komponentit, jotka mahdollistavat järjestelmän todellisen käytön. Kun turvajärjestelyt ovat kunnossa ja testilaitteisto on aseteltu toimimaan kyseisessä käytössä, voidaan työssä kehitetyn automaattitestilaitteiston luotettavuus todeta varmemmin.

Tulevaisuudessa automaattitestilaitteistossa on useita kehityssuuntia. Järjestelmää kannattaa kehittää nykyisestä, jotta se hyödyttäisi testausta enemmän. Tärkeitä kehityskohteita ovat testilaitteiston asettelun yksinkertaistaminen ja asetusmahdollisuuksien laajentaminen sekä testijärjestelyn virtaviivaistaminen.

Testin asettelussa täytyy tällä hetkellä asetella HTR testilaitteiston kommunikoinnin, tehon ja tyypin mukaiseksi ja määritellä testisekvenssi erilliseen tiedostoon sekä testata tämän kaiken toiminta. Lisäksi testissä käytetyt mittalaitteet täytyy asetella käsin niin, että AC500:n mittausten tallennus onnistuu halutusti. HTR:n asettelun vaikeus todennäköisesti ratkeaa, kun järjestelmää aletaan käyttää enemmän. Määrittelytiedostot täytyy asetella testilaitteistokohtaisesti kerran kohdalleen, jonka jälkeen samoja tiedostoja voi käyttää pienin muutoksin.

AC500:n asettelu puolestaan on asia, joka vaatii kehitystä. Tällä hetkellä AC500:n toiminta perustuu kiinteisiin ja käsin aseteltaviin mittausdatan muotoihin, joista poikkeaminen aiheuttaa ongelmia tallennuksessa. Logiikkaohjelmaan olisi hyvä kehittää HTR:n asetustiedostoa vastaava keskitetty sijainti, josta ohjelma lukee mitattavat kanavat ja niiden paikat. Ohjelma tulisi muokata lukemaan tällaista tiedostoa ja toteuttamaan mittausten tallentamisen sen mukaisesti. Tämä vaatii asetustiedoston lukemisen toteuttavan ohjelman osan, mittalaitekommunikoinnin muokkaamisen sellaiseksi, että se sisältää mittalaitteiden asettelun sekä mittausdatan tallentamisen muuttamista kanava-asetuksiin reagoivaksi. Mittalaitteiden kommunikointirajapinnat tarjoavat tällaiseen etäasetteluun lukuisia komentoja. AC500:ssa on useita tiedoston lukemiseen tarkoitettuja kirjastoja, mutta näiden tukemat tiedoston muodot täytyy selvittää ja tiedostosta lukeminen toteuttaa.

Tulevaisuutta ajatellen HPD:lla täytyy tehdä ratkaisu, miten automaattitestausta kehitetään pidemmällä aikavälillä. HTR:n ja AC500:n yhteiskäyttö on normaaliin mittaukseen melko kömpelö ratkaisu, sillä se vaatii kahden erillisen järjestelmän yhteistoimintaa. Tässä ratkaisussa HTR ei pysty hyödyntämään ohjauksessa kaikkia saatavilla olevia mittaustuloksia ja AC500 ei pysty ohjaamaan testiä saamiensa tulosten perusteella. Kaikki laitteistonvälisen kommunikoinnin muokkaaminen pitäisi tällä hetkellä toteuttaa sekä HTR:n ja taajuusmuuttajan, AC500:n ja taajuusmuuttajan, että HTR:n ja AC500:n välillä. Tällainen järjestely on herkkä muutoksista aiheutuville vioille.

Järkevä pidemmän aikavälin suunnitelma on keskittyä kehittämään jompaakumpaa mittaustapaa: HTR:ää tai AC500:aa. HTR:n puutteet ovat tällä hetkellä mittalaitetuen puuttuminen muilta kuin yhdeltä 34972A:lta, joka sekin täytyy kytkeä USB-piuhan välityksellä. Tämän takia HTR ei pysty käyttämään esimerkiksi todellisia sähköisiä suureita testien ohjaamiseen, eikä pysty valvomaan useampaa kuin 60 lämpömittauskanavaa. Lisäksi HTR:n käyttöliittymä on hieman kömpelö ja vaatii ohjelman sisäisen toiminnan tuntemusta. AC500 puolestaan kerää laajasti mittausdataa ja sillä on toteutettavissa datankäsittely sekä taajuusmuuttajakommunikointi kenttäväylätoteutuksena tai suoraan digitaalisilla ja analogisilla porteilla. AC500:aan täytyisi kuitenkin ohjelmoida käyttöliittymä ja taajuusmuuttajan ohjaus eri tilanteissa alusta alkaen, ja vaatisi paljon työtä ennen kuin pelkästään AC500:lla toteutettu testausjärjestely olisi HTR:n tämänhetkisellä tasolla.

Yksi mahdollisuus automaattitestauksen kehittämiseksi HPD:lla on siis ryhtyä HTR:n sidosryhmäksi ja käyttää resursseja testausympäristön kehittämiseen. Tällä hetkellä ympäristöä on kehitetty pelkästään LAC:n tarpeiden mukaan, mistä suuri osa sen puutteista johtuu. Jos HTR:n kehitystä jatkettaisiin myös HPD:n tarpeiden mukaisesti ja HPD tarjoaisi kehitykseen resursseja, voisi testausympäristössä olla selkeästi potentiaalia itsenäiseen toimintaan ilman toimintaa tukevia irrallisia komponentteja.

Toinen mahdollisuus on alkaa kehittää AC500:lla toteutettua testinohjausta, jolloin testien ohjaus tapahtuisi kenttäväylän yli, ja siinä käytettäisiin nyt toteutettua mittausdatan keräystä ohjaukseen. Tällä tavalla testin ohjaaminen tapahtuisi ehkä lopulta yksinkertaisemmin kuin nykyään, mutta taajuusmuuttajan parametrien ja sisäisten muuttujien lukeminen olisi nykyistä vaikeampaa ja pitäisi toteuttaa uudella tavalla. AC500:n lisäksi ulkoiseen testinohjaukseen on olemassa myös paljon muita teknologioita, joiden etuja AC500:n nähden voisi tässä tapauksessa tutkia.

Ohjelmoitavalla logiikalla toteutetun testausjärjestelyn kehittämisen etu HTR:n kehittämiseen verrattuna olisi kehitystyön keveys. HTR:n kehitystyö tapahtuu HPD:tä laajemmalla tasolla, ja sillä on useita sidosryhmiä joiden välistä kommunikointia muutosten toteuttaminen vaatii. Kehitystyöhön liittyminen vaatisi eri osastojen välistä sopimista ja kompromisseja sekä todennäköisesti ohjelmointiresurssien tarjoamista ympäristön kehittämiseen. Tästä irrallinen testinohjaus voitaisiin suorittaa projektina, jonka ainoa asiakas olisi HPD ja vaatimukset tulisivat yhdestä lähteestä. Tällä tavalla järjestelmän kehitys saataisiin ajan myötä HTR:n kehitystä nopeammaksi, ja uusi testausjärjestely voitaisiin suunnitella täysin omien tarpeiden mukaan. Tässä tapauksessa kestäisi kuitenkin suhteellisen pitkään, ennen kuin uusi testausjärjestelmä olisi tarpeellisella tasolla itsenäiseen toimintaan.

Miten tahansa automaattista testilaitteistoa kehittääkin, on yksi HPD:lle ominainen käyttö erilliset verkkopuolen moduulit ja niiden oma ohjelmisto. ISU-ohjelman ohjauksen liittäminen automaattiseen testilaitteistoon on parannus, jolla testijärjestelyä saadaan kohdennettua erityisen hyvin nimenomaan HPD:n käyttötarkoitukseen. Verkkopuolen ohjauksen lisääminen testiympäristöön parantaa testin mukautuvaisuutta huomattavasti ja mahdollistaa ISU-moduulien testauksen automaattisesti. Tämä on HPD:n tapauksessa oleellista.

LAC:ssa kestotestilaitteisto lähettää mittaustulokset reaaliajassa tietokantaan, josta ne ovat tarkasteltavissa halutussa muodossa ilman erillistä mittaustulosten keräämistä ja muotoilua. Tämä on asia, johon kannattaa tulevaisuudessa pyrkiä myös HPD:n mittaustulosten käsittelyssä. Tietokantayhteyden suunnittelussa aluksi selvitettäviä asioita ovat tietokannan perustamisesta ja käytöstä aiheutuvat kustannukset, mahdolliset rajoitukset datan määrän suhteen ja rajoitukset yhteyden datansiirtonopeudessa. LAC:lla tällä hetkellä käytetyssä tietokantayhteydessä reaaliajassa siirrettäviä mittauksia on alle 20, kun puolestaan HPD:n yleisessä testauskäytössä ei pelkästään yli 60 lämpömittausta ole harvinaista. Jos yhteyttä ei saada helposti suunniteltua sopivaksi yleisimpiin tilanteisiin, jää sen tuoma hyöty pieneksi.

10.2 Järjestelmätestauksen kehittäminen

Jos testaushenkilöstöä saadaan vapautettua testinvalvontatehtävistä ja testaus saadaan toteutettua nykyistä tehokkaammin, on vapautuvia testausresursseja hyödyllistä käyttää testauksen kehittämiseen. Testaushenkilöstö voi mahdollisuuksien mukaan luoda uusia testitilanteita havaittujen puutteiden ja kokemuksen perusteella, ja samalla toteuttaa tarpeelliseksi havaittuja testejä. Tässä aliluvussa esitetään muutamia vaihtoehtoja järjestelmätestauksen kehityssuunniksi.

Helpoin ensiaskel automaattisen testauksen tuottaman testausajan hyödyntämiseksi on lisätä kuormittavan testauksen määrää ja kattavuutta nostamalla itsenäiseen testaukseen hyväksyttyjen testipaikkojen testiaikaa mahdollisimman suureksi ja lisäämällä eri toimintapisteitä automaattisiin kuormitustesteihin. Jo pelkästään testausaikaa lisäämällä on mahdollista löytää piileviä ja satunnaisia vikoja testattavien laitteiden toiminnassa ja luotettavuudessa, ja testipaikkoja lisäämällä saadaan testattavien taajuusmuuttajien kohtaamaa kuormitusta monipuolistettua. Jokin taajuusmuuttajan komponentti saattaa pettää vain harvinaisissa olosuhteissa tai pidemmän ajan kuluttua, ja tällaisia ongelmia on mahdollista saada kiinni pitkillä testisekvensseillä. Kiihdytettynä kestotestauksena tällainen testaustapa on vielä tehokkaampi.

Monien alijärjestelmien kannalta haastavimpia käyttötapoja ovat dynaamiset tilanteet kuten kuorman muutokset, kiihdytykset ja jarrutukset. Myös huono maadoitus tai maasulkutilanteessa testaaminen on raskasta eri alijärjestelmille. Tällaisen testauksen lisääminen nykyisestä joko automaattisesti tai valvottuina testeinä on toteutettavissa suhteellisen helposti. Esimerkiksi ramppiaikojen toteutumisen seuraaminen ja momentin tarkkuuden mittaaminen eri tilanteissa ovat suorituskykytekijöitä, jotka ovat taajuusmuuttajalle tärkeitä ominaisuuksia. Momentin laadulliseen testaukseen vaaditaan momenttiantureita testipaikoille, mutta muuten dynaamisten tilanteiden testaaminen vaatii lähinnä testinsuunnittelua ja testiaikaa.

Taajuusmuuttajan toiminnallisen testauksen laajentaminen vaatii nykyisen testausohjelman tarkkaa vertaamista eri järjestelmän toimintaa kuvaaviin dokumentteihin puutteiden havaitsemiseksi. Järjestelmätestausta kehittääkseen pitää suunniteltavia testejä ohjata koko järjestelmän toimintaa koskevien käyttötilanteiden testaukseen sekä järjestelmän ja ympäröivän maailman rajapintaan. Toiminnallisen testauksen kattavuutta voidaan tutkia perehtymällä taajuusmuuttajasukupolven määritteleviin dokumentteihin sekä käyttäjädokumentaatioon ja käymällä ominaisuus ominaisuudelta läpi järjestelmän toiminnalle suunnitellut käyttötavat, raja-arvot ja vianhallinta ja priorisoimalla tämän tutkinnan perusteella lisää testitapauksia läpikäytäviksi. Tämän tutkimuksen seurauksena löytyvistä tilanteista voidaan muodostaa mustan laatikon testausmenetelmillä uusia toiminnallisia testejä. Toiminnallisesta testauksesta paljastuvia ongelmia selvitettäisiin paljastumisen jälkeen testaus- ja suunnitteluyksiköiden yhteistyöllä mustan ja valkoisen laatikon menetelmiä yhdistämällä.

Toiminnallisen testauksen automatisointi joissakin rutiinitesteissä voi olla mahdollisuus tehostaa testausta, jos pystytään kehittämään tarpeeksi laaja, monipuolinen ja useista toistoista hyötyvä testisekvenssi. Toiminnallisen testauksen vaatiman ohjauksen voi toteuttaa ohjelmoitavan logiikan digitaalisilla ja analogisilla porteilla. Sisään- ja ulostuloihin sekä signaaliteihin kohdistuvia testejä on jo nykyisessä testiohjelmassa useita, mutta niiden toiston tarve ja tiheys täytyisi arvioida, jotta voidaan päättää olisiko automatisointi hyödyllistä. Jos toiminnallisen testauksen automatisointi koetaan hyödylliseksi, voi testirutiineja toteuttaa esimerkiksi AC500-logiikalla. Yksi toiminnallisen automaattitestauksen kohde saattaa olla järjestelmätason regressiotestausrutiini.

Testauksen suunnittelun yksi tärkeä tavoite on, että suunnitellut testaustoimenpiteet ovat hyödynnettävissä mahdollisimman pitkään. Tämän edistämiseksi kaikki nykyiset ja uudet testit olisi ajan kuluessa suotavaa määritellä ja ohjeistaa yhdenmukaisella tavalla, jotta niiden toteutus eri testaajien toimesta eri aikoina ja jopa eri tuotteilla on keskenään mahdollisimman vertailukelpoista. Tällaiseen kehitykseen kuuluisi perusteellisesti suunniteltu testiohjelma, jossa jokaiselle testille on määritelty muun muassa vaatimukset, olosuhteet, käytetty laitteisto, testin kulku ja vaaditut mittaukset sekä raportointitapa. Tällainen testisuunnitelma vähentäisi riskiä, että tulevaisuudessa vertaillaan testituloksia aikaisemmin eri tavalla toteutettuihin vertailukelvottomiin testeihin tai jonkin testin toisto epäonnistuu myöhemmin eri yhteydessä toteutettuna. Myös asioiden uudelleenopettelua henkilöstön vaihtuessa ja ajan kuluessa saataisiin vähennettyä perusteellisella testausdokumentaatiolla.

11 Yhteenveto

Tässä diplomityössä tutkittiin järjestelmätestauksen roolia tuotekehityksen testauksessa ja tutkittiin eri menetelmiä, joita järjestelmätestausvaiheessa voidaan hyödyntää. Tämän perusteella yritettiin kehittää ABB High Power Drivesin järjestelmätestausta. Järjestelmätestauksen tehostamiseksi toteutettiin automaattitestilaitteisto, jolla tulevaisuudessa on tarkoitus vapauttaa henkilöstöresursseja pitkäkestoisista rutiinitestiajoista muuhun testaukseen. Testausteorian, taajuusmuuttajan ominaisuuksien ja eri osa-alueiden asiantuntijahaastatteluiden perusteella tutkittiin mahdollisia tapoja hyödyntää testausautomaatiota järjestelmätestauksessa ja laajentaa testien kattavuutta automaation myötä lisääntyvän testausajan avulla.

Työn aikana toteutettu automaattitestilaitteisto suunniteltiin, toteutettiin ja koekäytettiin eri tilanteissa. Laitteisto todettiin toimivaksi, mutta se vaatii vielä pidempiaikaista käyttöä toiminnan varmistamiseksi. Työssä pohdittiin kehitetyn testilaitteiston puutteita ja mahdollisia tapoja kehittää laitteistoa nykyisestä.

Järjestelmätestauksen kehittämiseksi ehdotettiin työssä useita eri mahdollisuuksia, joista osa on toteutettavissa automaattitestilaitteiston testisekvenssejä suunnittelemalla ja osa manuaalisella testauksella tai automaattitestilaitteiston toimintaa kehittämällä. Työssä pohditun järjestelmätestauksen kehittämisen tavoitteena on tehostaa testausta automatiikalla, jonka vapauttamia resursseja voidaan käyttää testien kattavuuden lisäämiseen uusia testejä suunnittelemalla ja toteuttamalla. Samalla automaattitestijärjestelyjä voidaan kehittää työn jälkeen helppokäyttöisemmäksi ja monipuolisemmaksi.

Testauksen jatkuva kehittyminen on oleellista suunniteltavien järjestelmien monimutkaistuessa ja järjestelmille asetettujen vaatimusten kasvaessa. Tämän takia kaikki testauksen osa-alueet vaativat jatkuvaa kehittymistä siinä missä järjestelmäsuunnittelukin. Tässä diplomityössä on pyritty kehittämään järjestelmätestausvaihetta. Ongelmana on testaustarpeen kasvaminen testausresurssien säilyessä suunnilleen aikaisemmalla tasolla. Kaikki ajatukset testauksen tehostamiseksi ovat tämän takia tarpeellisia. Automaattitestaus on yksi tapa muiden joukossa vapauttamaan henkilöstöä muuhun käyttöön ja lisäämään käytettävissä olevaa testausaikaa. Onnistuessaan hyvin automaattitestaus mahdollistaa testauksen määrän lisäämisen, mikä parantaa suunnitellun järjestelmän laatua. Epäonnistuessaan automaattitestaus vaatii ylläpitoonsa enemmän resursseja kuin se vapauttaa. Tämän takia testaus pitää suunnitella automaattisesti ja manuaalisesti toteutettavia testejä yhdistelemällä, niin että molempien testaustapojen parhaat puolet hyödynnetään. Tällainen testauksen suunnittelu on pitkä ja vaativa prosessi, jota jatkuvasti tekemällä pystytään kuitenkin varmistamaan suunnitellun tuotteen laatu nykyhetken lisäksi myös tulevaisuudessa.

Lähdeluettelo

[1] A. Engel, *Wiley Series in Systems Engineering and Management: Verification, Validation and Testing of Engineered Systems*, A. P. Sage, Ed. Wiley: Hoboken, NJ, USA, 2010, pp. 3-149, 351-487.

[2] G. J. Myers, C. Sandler and T. Badgett, *Art of Software Testing*, 3rd edition, Wiley: Hoboken, NJ, USA, 2012, pp. 5-18, 41-84, 113-142.

[3] IEEE, *IEEE 829-2008: IEEE Standard for Software and System Test Documentation,* 2008, pp. 11.

[4] R. Black, *Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing*, 3rd edition. R. Elliot, K. Talbot, J. McKay, D. Scribner and C. English, Eds. New York: Wiley, 2009, pp. 1-49, 79-145, 553-567.

[5] E. A. Elsayed, "Models for accelerated life testing" in *Reliability Engineering*, 2nd edition, A. P. Sage, Ed. Hoboken, N.J., Wiley, 2012, pp. 364-440.

[6] D. Morgan, *A Handbook for EMC Testing and Measurement*, Stevenage : The Institution of Engineering and Technology, 2007, pp. 1-48.

[7] ABB Drives, "Technical guide no. 3: EMC compliant installation and configuration for a PDS, technical guide no. 4: Guide to variable speed drives" in *Technical Guide Book*, Revision H, 2014, pp. 116, 125-127, 158-165.
Saatavilla: ABB Download Center,
http://www.abb.fi/abblibrary/DownloadCenter/Default.aspx

[8] ABB Drives, *ABB Industrial Drives ACS880, Multidrives 1.5 to 5600 kW Catalog*, Revision D, 2014, pp. 25-27.
Saatavilla: ABB Download Center,
http://www.abb.fi/abblibrary/DownloadCenter/Default.aspx

[9] ABB Drives, *ACS880 Primary Control Program Firmware Manual*, Revision I, 2014, pp. 30.
Saatavilla: ABB Download Center,
http://www.abb.fi/abblibrary/DownloadCenter/Default.aspx

[10] ABB Drives, *User's Manual - FSO-12 Safety Functions Module*, Revision A, 2014, pp. 40.
Saatavilla: ABB Download Center,
http://www.abb.fi/abblibrary/DownloadCenter/Default.aspx

[11] Kidde AirSense, *Stratos Micra 25 Aspirating Smoke Detector Datasheet,* 2013.

[12] Agilent Technologies, *Agilent 34970A/34972A Data Acquisition / Switch Unit User's Guide*, 3rd Edition, 2012, pp. 5, 12.

[13] Yokogawa Electric Corporation, *WT3000 Precision Power Analyzer User's Manual*, 5th Edition, 2007, pp. 30, 79.

[14] ABB Drives, *Heat Test Runner Quick Guide.* 2013. Sisäinen dokumentti.

[15] ABB Industrial Automation & Motion, *Industrial Automation & Motion PLCs, HMIs, Drives, Servo Drives, Motion Controllers Main Catalog*, 2014, pp. 9, 11, 14-15.
Saatavilla: ABB Download Center,
http://www.abb.fi/abblibrary/DownloadCenter/Default.aspx

[16] *Codesys-ohjelmointiympäristön verkkosivu*. Vierailtu 11.09.2014.
Saatavilla: http://www.codesys.com/.

[17] ABB, PS501 Control Builder Plus v2.3.0 Codesys -käyttäjädokumentaatio. 2013.